Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

PROGETTAZIONE E SVILUPPO DI UNA PIATTAFORMA PER ECOSISTEMI DI DIGITAL TWIN

Elaborato in:
Pervasive Computing

Relatore:
Prof. Alessandro Ricci
Correlatori:
Prof. Marco Picone
Dott. Samuele Burattini

Presentato da:
Marta Spadoni

Sessione IV
Anno Accademico 2021-2022

Indice

Introduzione

Una delle tecnologie che nell'ultima decade ha attirato maggior interesse, sia a livello accademico che industriale, è quella rappresentata dai Digital Twin (DT). Con il termine Digital Twin si fa riferimento alla possibilità di creare un clone software di una qualsiasi entità fisica: macchinari industriali, oggetti comuni, veicoli aerospaziali ma anche entità intangibili come processi e attività, con lo scopo di poter monitorare in real-time lo stato dell'oggetto reale e utilizzare la copia digitale per fornire ulteriori servizi [1].

Il paradigma ha trovato applicazione in molteplici ambiti, da quello industriale all'*healthcare* passando per le *smart city*, ma questa grande e repentina diffusione ha portato ad un'assenza di consenso generale circa la definizione e le proprietà del concetto stesso.

La mancanza di uniformità nella modellazione del concetto ha determinato lo sviluppo di molteplici soluzioni software ognuna con la propria visione del paradigma, causando così scarsa interoperabilità. L'assenza di Application Programming Interface (API) comuni e standard ha generato dei *silos* software per mezzo dei quali risulta estremamente difficile, se non impossibile, generare ecosistemi di Digital Twin attraverso i quali sfruttare tutte le potenzialità del paradigma [1].

In contrapposizione alla visione fornita attualmente dalla letteratura, Web of Digital Twins (WoDT) estende il paradigma con lo scopo di virtualizzare complesse realtà di asset interconnessi, appartenenti anche a domini e organizzazioni differenti, nell'ottica di creare degli *open-system* [2]. Nello specifico, seguendo l'approccio WoDT è possibile definire ecosistemi aperti, distribuiti e dinamici costituiti da Digital Twin connessi tra loro, i quali possono essere utilizzati in modalità *as-a-service* senza essere legati esclusivamente ad applicazioni specifiche [3] [2].

Il presente progetto di tesi ha l'obiettivo di progettare e sviluppare una

prima piattaforma per supportare ecosistemi di Digital Twin. Al fine di implementare un *Web of Digital Twins*, oltre alla piattaforma, è necessario poter definire i singoli DT come delle entità attive che espongono una precisa API con lo scopo di garantire l'interazione dinamica con le applicazioni attraverso i protocolli propri del Web. Dunque, due ulteriori obiettivi di questa tesi sono la realizzazione di un framework per generare DT aderenti al modello proposto in WoDT e la definizione del concetto di *Digital Twin Descriptor*.

Il presente documento è suddiviso in cinque capitoli, i primi due consentono di presentare al lettore il contesto teorico in cui si colloca il progetto, mentre i rimanenti riportano il processo di analisi, progettazione e sviluppo seguito. Nello specifico, il primo capitolo descrive lo stato dell'arte del paradigma Digital Twin, presentando la sua storia, le diverse definizioni presenti in letteratura, gli ambiti di applicazione e le principali problematiche. Il secondo capitolo affronta un'analisi del concetto di Web of Digital Twins, descrivendo i suoi obiettivi e le caratteristiche che lo contraddistinguono. Successivamente, il terzo capitolo presenta i requisiti per la realizzazione della piattaforma, il quarto descrive la fase di progettazione dell'architettura della piattaforma e dei singoli Digital Twin e l'ultimo fornisce i dettagli implementativi delle diverse componenti software realizzate nel contesto di questo elaborato e un esempio d'uso di quanto realizzato.

Capitolo 1

Digital Twin: lo stato dell'arte

In questo primo capitolo sono, inizialmente, presentati la storia e lo sviluppo del concetto di Digital Twin (DT), dalla prima formulazione alla sua attestazione come tecnologia emergente e fondamentale per il futuro.

Di seguito, vengono discusse le diverse definizioni del concetto presenti allo stato dell'arte e le principali caratteristiche del paradigma.

Successivamente, sono presentati i principali ambiti in cui i DT hanno trovato una naturale applicazione e sono analizzate le tecnologie più diffuse e utilizzate per implementare sistemi che li impiegano.

Infine, sono delineate le problematiche più importanti che caratterizzano il nuovo paradigma.

1.1 Storia

L'origine del concetto di Digital Twin (DT) risale al 2002, anno in cui Michael Grieves presenta "un ideale concettuale" per il Product Lifecycle Management (PLM) durante una presentazione per la formazione di un centro di PLM [4] [5]. Tale modello concettuale fu ripreso nel 2003, dallo stesso Grieves, nel corso di PLM all'Università del Michigan dove viene definito *Mirrored Space Model*. Sebbene Grieves non utilizzi il termine Digital Twin, il concetto delineato presenta le tre caratteristiche principali del paradigma: un oggetto nello spazio fisico, una rappresentazione nello spazio virtuale e una connessione tra i due spazi per il flusso di dati [4] [5]. Solo nel 2014, Grieves formalizza, nel whitepaper " Digital Twin: Manufacturing Excel-

lence through Virtual Factory Replication" [5], il concetto di Digital Twin definendone i punti chiave. Quest'ultimi saranno analizzati nel paragrafo 1.2.

L'idea di sfruttare dei sistemi gemelli per progettare, monitorare e trovare la soluzione a eventuali problemi è utilizzata dalla NASA già dal 1970 [6]. L'esempio più famoso è, probabilmente, quello della missione Apollo 13. In questo caso, fu costruito un complesso sistema di simulatori utile sia per addestrare gli astronauti prima della partenza che per replicare e monitorare le condizioni del veicolo nello spazio durante la missione [7] [6]. Grazie alla possibilità di simulare le condizioni delle varie componenti dell'Apollo 13, inserendo i dati provenienti dallo spazio nei diversi simulatori, è stato possibile determinare come risolvere il danno causato dall'esplosione di una tanica di ossigeno avvenuta due giorni dopo il lancio [8].

La volontà di ampliare la conoscenza del concetto di sistemi gemelli e il desiderio di ridurre i costi e le risorse impiegate, hanno portato la NASA allo studio e allo sviluppo di Digital Twin per i loro veicoli spaziali [6]. Nel 2010, nella sua *Roadmap*, la NASA definisce i Digital Twin come lo strumento principale per l'ingegnerizzazione dei loro sistemi di simulazione e per migliorare le performance nel campo dell'aviazione [9] [6].

Quando il concetto di realizzare dei gemelli virtuali di prodotti fisici fu introdotto nel 2003, le tecnologie a disposizione per raccogliere i dati necessari a modellare una versione digitale di un oggetto fisico e, in generale, per implementare tale software erano limitate e immature [5]. L'impossibilità di realizzare concretamente il modello proposto ha portato la comunità scientifica stessa a non esplorare le potenzialità dei Digital Twin. Si nota, infatti, che dal 2003 al 2011 gli articoli riguardanti i DT e concetti simili sono quasi inesistenti [10].

Il rapido sviluppo delle tecnologie di comunicazione, la diffusione dell'Internet of Thing (IoT), e dunque dei Big Data, e l'evoluzione delle tecnologie di simulazione e dell'Intelligenza Artificiale hanno contribuito all'esplosione dei Digital Twin fornendo un supporto maturo e pratico alla loro implementazione [10]. Dal primo *paper* del 2011, il quale trova una concreta e utile applicazione dei DT nel campo dell'aviazione, e dalla formalizzazione del concetto da parte della NASA nel 2012 e da Grieves nel 2014, l'interesse scientifico per questa tecnologia è aumentato esponenzialmente [10] [6].

Oltre al notevole aumento di articoli scientifici relativi ai DT, che trovano

una possibile applicazione del paradigma in molteplici ambiti, dal 2017 per tre anni consecutivi Gartner classifica i Digital Twin come una delle top 10 tecnologie strategiche dell'anno [11] [12] [13]. Nel 2020, inoltre, Gartner dichiara i DT come una delle tecnologie emergenti che nei successivi 5-10 anni avranno un significativo impatto sulla società e l'economia [14].

1.2 Definizioni e caratteristiche

La repentina crescita dell'interesse scientifico nei confronti dei Digital Twin, avvenuta negli ultimi dieci anni, ha portato al proliferare delle definizioni del termine, ognuna delle quali fornisce la propria visione del concetto a seconda dell'ambito di applicazione. In linea generale, le diverse definizioni concordano, però, con l'idea di replicare, attraverso un software, un oggetto fisico. Ciò che, invece, le contraddistingue è l'insieme delle proprietà e delle caratteristiche che individuano per il concetto [1].

La prima vera definizione del termine Digital Twin viene riportata della NASA nella sua *Roadmap* [9] del 2010:

> *A Digital Twin is an integrated multiphysics, multiscale simulation of a vehicle or system that uses the best available physical model, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin.*

Concordemente all'ambito in cui la National Aeronautics and Space Administration opera, la definizione proposta si concentra su una visione del termine legata alla ricerca aerospaziale, uno dei primi settori, insieme a quello manifatturiero, in cui il paradigma ha trovato applicazione [6].

Lo stesso Grieves, nel 2017 in "Digital Twin: Mitigating Unpredicatable, Undesirable Emergent Behavior in Complex Systems" [15] fornisce la sua definizione:

> *The Digital Twin is a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level. At its optimum, any information that could be obtained from inspecting a physical manufactured product can be obtained from its Digital Twin.*

La definizione riportata non descrive l'oggetto fisico virtualizzato come un'entità qualsiasi del mondo reale bensì come un prodotto industriale, questo è dovuto al fatto che Grieves concepì il concetto di DT proprio in relazione al settore manifatturiero. Un aspetto importante che, invece, emerge dalla definizione analizzata, è la possibilità di utilizzare il Digital Twin come strumento per ottenere informazioni dirette circa lo stato dell'oggetto fisico.

L'applicazione del paradigma in ambiti diversi da quello industriale e dell'aviazione ha trasformato la definizione dal riferirsi unicamente ad artefatti industriali all'includere un qualsiasi oggetto fisico potenzialmente anche intangibile [1], come processi e attività. Minerva *et al.* in [1] forniscono una nuova e più generale definizione:

> *A Digital Twin is a comprehensive software representation of an individual physical object. It includes the properties, conditions, and behavior(s) of the real-time object through models and data. A Digital Twin is a set of realistic models that can simulate an object's behavior in the deployment environment. The Digital Twin represent and reflects its physical twin and remains its virtual counterpart across the object's entire lifecycle.*

Tale definizione racchiude le 3 caratteristiche fondamentali del paradigma [1] [6]:

1. Un DT si riferisce a un oggetto fisico modellandone proprietà e comportamento.

2. Un DT permette di modellare un oggetto fisico per tutto il suo ciclo di vita, evolvendosi insieme a esso. In questo modo, il DT permette anche di descrivere la storia dell'*asset* fisico.

3. Un DT contiene tutte le informazioni necessarie per predire e simulare il comportamento del relativo oggetto fisico.

La capacità di un Digital Twin di modellare e descrivere adeguatamente gli aspetti chiave dell'*asset* fisico è essenziale per rappresentare a livello digitale le sue proprietà e il suo comportamento [10] senza includere dettagli non necessari [5] ma consentendo, allo stesso tempo, di simulare, predire e ottimizzare il comportamento dell'oggetto fisico. La modellazione del DT

deve selezionare le proprietà della componente fisica a diverse scale e livelli, includendo quindi sia proprietà macroscopiche come forma e dimensione ma anche caratteristiche microscopiche come ad esempio la ruvidezza del materiale della sua superficie [16] [9] [4].

In "Digital Twin modelling" [17] Tao *et al.* scrivono che il modello scelto per il Digital Twin è uno dei suoi componenti fondamentali al punto da determinare la possibilità di utilizzare con successo o meno il DT stesso. Il modello costruito si basa su 4 dimensioni: quella geometrica, quella fisica, quella comportamentale e quella delle regole. La prima componente descrive per l'appunto la forma geometrica e le modalità di costruzione dell'entità fisica. Il modello fisico, invece, riflette le proprietà, le caratteristiche e i vincoli fisici che definiscono l'oggetto da modellare. Il modello di comportamento costituisce la componente dinamica e modella come l'entità fisica risponda a meccanismi ed eventi interni ed esterni. In ultimo, il modello delle regole permette al Digital Twin di incorporare e elaborare i dati storici al fine di estrapolarne conoscenza, utilizzando, ad esempio, anche opportune ontologie per comprendere e formalizzare i dati [6].

L'idea che il Digital Twin non sia una mera rappresentazione statica della relativa entità fisica ma che, al contrario, sia un modello dinamico che evolve insieme al suo oggetto fisico per tutto il ciclo di vita, è presente già dalla prima formulazione concettuale di Grieves del 2002 [4]. L'evoluzione della copia virtuale è possibile grazie alla continua interazione, comunicazione e sincronizzazione con l'oggetto fisico che viene realizzata per mezzo della relazione bidirezionale presente tra i due [6].

L'insieme delle caratteristiche del processo che lega la copia virtuale all'entità fisica è uno dei punti su cui la letteratura scientifica è meno allineata. In generale, si concorda che la connessione tra il piano digitale e quello fisico sia abilitante per il concetto di Digital Twin, in quanto, permette a quest'ultimo di progredire e di allinearsi costantemente al suo oggetto fisico per tutto il suo ciclo di vita [5] [1] [6]. Si dibatte invece, su alcune proprietà del legame, come ad esempio, la direzione del flusso delle informazioni e il suo *rate* di aggiornamento. Per quanto riguarda il primo aspetto, si riconosce, in generale, la necessità che le informazioni debbano fluire dal livello fisico a quello digitale ma per molti può e deve sussistere anche il flusso contrario. Quest'ultimo infatti, fornisce al DT la possibilità di inviare comandi ottimizzati

all'oggetto fisico [6] [1]. Relativamente alla velocità con cui le informazioni vengono scambiate tra la copia digitale e l'entità reale, Minerva *et al.* in [1] forniscono una visione pratica: il legame tra l'oggetto digitale e quello fisico è efficace se il tempo di aggiornamento dell'oggetto logico è minore del tempo medio di accesso alle informazioni da parte delle applicazioni che utilizzano il DT.

Inoltre, la componente dinamica presente nella natura del Digital Twin, che non lo rende unicamente un *mapping* statico delle proprietà di un *asset* fisico [18], si traduce nella capacità del DT stesso di simulare il comportamento dell'oggetto fisico. Questa capacità, considerata una delle qualità principali del paradigma [18], consente alla copia virtuale di prediFre il comportamento dell'entità fisica in determinate situazioni e ambienti, al fine di anticipare e prevenire problemi ed errori [1]. Integrando algoritmi di Intelligenza Artificiale (AI), il Digital Twin acquisisce inoltre l'abilità di scoprire informazioni, schemi nascosti e correlazioni sconosciute [6].

1.2.1 Proprietà dei Digital Twin

Grieves, nel suo *whitepaper* [5] di formalizzazione del concetto di Digital Twin, riconosce nel paradigma 3 attributi propri dell'essere umano:

Contestualizzazione Una capacità peculiare dell'uomo è quella di poter visualizzare nell'insieme una situazione, capire il contesto e concettualizzare un problema.

Nell'analizzare una problematica, tipicamente, gli umani tendono a utilizzare la propria vista per ridurre il problema in numeri, simboli e lettere per poi contestualizzarli nuovamente attraverso la vista. Questo doppio passaggio porta ad una perdita di informazione ed è inefficiente.

La capacità dei Digital Twin di fornire in real-time la visualizzazione completa della situazione da analizzare e simultaneamente di indicare i parametri desiderati, e quelli futuri nel caso della simulazione, permette di contestualizzare e analizzare la problematica molto più velocemente.

Comparazione Uno strumento tipicamente utilizzato dagli umani per valutare una situazione è il confronto. In modo conscio, o anche inconsciamente,

l'uomo tende a comparare i risultati ottenuti con quelli desiderati, al fine di determinare la differenza e di capire come eliminarla o ridurla.

I Digital Twin consentono di visualizzare allo stesso tempo lo stato attuale dell'oggetto fisico e gli intervalli di valori entro cui ogni suo parametro dovrebbe rientrare. In questo modo, si velocizza il confronto tra la situazione attuale e quella desiderata consentendo di attuare le eventuali attività di *recovery* necessarie in modo più tempestivo.

Collaborazione Lo strumento più potente nelle mani degli uomini è sicuramente la collaborazione, consente di riunire più punti di vista e di trovare soluzioni più intelligenti e innovative ai problemi. Questo strumento si scontra però con il limite umano costituito dal fatto che la concettualizzazione o contestualizzazione del problema avviene unicamente nel singolo individuo.

I Digital Twin permettono di condividere la contestualizzazione di un problema, abilitando, infatti, un numero illimitato di persone che non necessariamente si trovano nello stesso luogo a visualizzare nel medesimo modo una stessa situazione.

Idealmente, nel mondo virtuale creato dai Digital Twin, la posizione fisica degli individui diventa irrilevante e gli esseri umani, in qualsiasi parte del mondo, possono avere una visuale comune del contesto, realizzare confronti per determinare eventuali discostamenti e collaborare insieme.

Minerva *et al.* nel loro *survey* [1] tentano di delineare una visione consolidata e uniforme del paradigma: oltre che fornire la definizione precedentemente riportata, illustrano un insieme di caratteristiche che contraddistinguono i Digital Twin, nello specifico in relazione con l'*Internet of Things*, concetto che da un lato trae beneficio dal paradigma stesso e dall'altro ne è un fattore abilitante.

Di seguito vengono descritte e analizzate le caratteristiche allo stato dell'arte individuate da Minerva *et al.*.

Rappresentatività Come in parte già precedentemente riportato, la copia virtuale deve essere quanto più possibile verosimile all'entità fisica di riferimento. Ciò nonostante rappresentare l'oggetto fisico nella totalità dei suoi aspetti può essere estremamente complesso e soprattutto inutile. Alcu-

ne delle caratteristiche dell'oggetto, infatti, possono non essere funzionali a raggiungere l'obiettivo per il quale si implementa il Digital Twin.

Il modello del Digital Twin deve essere progettato e implementato avendo chiaro il contesto in cui questo andrà ad operare e dovrà rappresentare, esclusivamente, le proprietà, le caratteristiche e i comportamenti dell'*asset* fisico che sono necessari e sufficienti a raggiungere i propri *goal* e a classificare l'oggetto virtuale come rappresentativo di quello fisico sotto tutti i suoi aspetti.

Riflessione Un Digital Twin deve essere in grado di riflettere lo stato del relativo oggetto fisico, ciascun cambiamento di stato e evento generato dall'entità fisica deve essere riportato dal DT [19].

Lo stato dell'oggetto fisico, inteso come l'insieme dei valori degli attributi, degli eventi e delle azioni, cambia rispetto al tempo, per questo motivo si può affermare che è stato adeguatamente descritto attraverso il Digital Twin, se ciascun valore è tempestivamente mappato nel suo equivalente nello stato gemello. La proprietà di riflessione si riferisce per l'appunto alla capacità del modello del DT di rappresentare univocamente, ma non necessariamente in rapporto uno-a-uno, ciascuna caratteristica rilevante dell'oggetto fisico.

Replicazione Le due precedenti proprietà determinano che il modello utilizzato per un DT, cioè l'insieme delle caratteristiche dell'entità fisica che si decide di considerare e modellare, deve essere quello minimo per rendere efficace ed efficiente l'utilizzo della copia virtuale nel contesto applicativo in cui deve operare. Questa necessità suggerisce che ciascun concetto reale, e ricorsivamente anche ciascuna copia software, possa essere virtualizzato e replicato più volte in diversi ambienti virtuali. Ciascun clone sarà modellato su misura per i requisiti applicativi che deve soddisfare.

Entanglement Fin dal concepimento del paradigma, questo comprende 3 elementi fondamentali: i) un oggetto fisico, ii) una copia digitale e iii) una connessione tra i due. Con il termine *entanglement* Minerva *et al.* si riferiscono proprio a questa terza componente, cioè al legame tra il livello fisico e quello virtuale attraverso cui fluiscono, in *real-time*, le informazioni

che descrivono la prima componente, in modo tale che il clone software possa elaborarle e renderle disponibili alle applicazioni.

Le proprietà dell'*entanglement* caratterizzano il legame tra l'oggetto fisico e virtuale, nello specifico come vengono scambiate le informazioni tra i due. Le 3 caratteristiche da considerare sono:

1. **Connettività**: L'*asset* fisico e la copia digitale devono poter comunicare i cambiamenti di stato in modo diretto o indiretto. Questo implica che l'oggetto reale debba essere dotato di un mezzo di comunicazione che consenta la trasmissione diretta delle informazioni o che debba esistere un mediatore (esterno) che possa, in sostituzione all'oggetto fisico, inviare e ricevere le informazioni al Digital Twin. In quest'ultimo caso, la terza entità deve essere in grado di osservare e monitorare lo stato della componente fisica, in modo tale da poter inviare i dati al DT. Ad esempio, il monitoraggio dello stato può avvenire per mezzo di sensori e videocamere.

2. **Tempestività**: Lo scambio di informazioni tra le due entità, quella fisica e quella digitale, deve avvenire in modo tempestivo o quanto meno deve richiedere un tempo trascurabile rispetto a quello che intercorre tra i cambiamenti di stato dell'oggetto fisico e all'utilizzo previsto per la copia digitale. Indicativamente, il tempo medio che intercorre tra due cambi di stato successivi di una delle due parti è da considerarsi come limite massimo per il tempo richiesto per l'invio delle informazioni all'altra componente.

3. **Associazione**: La relazione che lega l'entità logica e quella fisica può essere unidirezionale o bidirezionale. Nel primo caso, il flusso informativo può muoversi dal piano fisico a quello virtuale, come nel caso di sensori, o viceversa, dal piano digitale a quello reale, come nel caso di attuatori a cui inviare comandi. Nel secondo caso, invece, lo scambio di informazioni avviene in ambo le direzioni, ciò può portare ad un grande valore aggiunto per l'oggetto fisico, il quale può ricevere aggiornamenti e comandi mirati a migliorare il suo funzionamento.

Persistenza Questa proprietà si riferisce al fatto che un DT deve poter persistere nel tempo, cioè deve superare i limiti fisici a cui può essere sottoposto

l'oggetto immerso nel mondo reale e deve garantire una continua disponibilità alle applicazioni che vi si interfacciano. Inoltre, in caso di malfunzionamenti nell'oggetto fisico, la copia digitale deve essere la fonte di informazioni per ripristinarne lo stato.

Memorizzazione Un Digital Twin deve essere in grado di memorizzare e rappresentare tutte le informazioni, presenti e passate, che siano rilevanti. In relazione al concetto di rappresentatività e alla capacità di contestualizzazione che deve offrire un Digital Twin, la quantità di informazioni grezze o elaborate che è necessario gestire può non essere banale. In particolare, questi grandi *dataset* hanno un duplice utilizzo, da un lato consentono di capire e analizzare lo stato e il comportamento dell'oggetto fisico e dall'altro abilitano il DT a predire lo stato futuro dell'oggetto fisico, anche in eventuali differenti condizioni ambientali.

Compositività Gli oggetti reali sono spesso costituiti dall'aggregazione di più singole entità. In relazione a questo, un Digital Twin deve possedere l'abilità di monitorare contemporaneamente sia il comportamento dell'oggetto composito sia quello delle componenti.

Considerando un sistema di grandi dimensioni, ciascuna componente, o sotto-sistema, deve poter essere rappresentata da un clone digitale apposito; questo insieme di copie software devono però poter collaborare e interagire come se fossero un'unica grande entità per le applicazioni che lo necessitano.

D'altra parte, questa proprietà fa riferimento alla possibilità, per mezzo del DT, di astrarre dalla complessità di un sistema di grandi dimensioni, in modo tale da potersi concentrare sugli aspetti rilevanti e non dover considerare ogni singola componente.

Responsabilità e gestione Ciascun Digital Twin deve poter essere totalmente e accuratamente gestito, anche nel caso di malfunzionamenti. Dopo essere entrato in uno stato di *recovery*, deve poter continuare a rispondere ad interrogazioni circa lo stato della controparte fisica, fornendo le ultime informazioni rilevate e disponibili.

Come riportato precedentemente, un Digital Twin deve rappresentare l'entità fisica per tutto il suo ciclo di vita e non solo. Questa proprietà, infatti,

determina che la copia virtuale debba continuare ad esistere ed operare anche oltre il tempo di "vita" dell'entità fisica.

La possibilità di applicare il paradigma in diversi ambiti, come ad esempio anche quello del *healthcare*, determina, inoltre, la necessità di avere per lo stesso DT diversi livelli di gestione e responsabilità.

Augmentation Contrariamente a quanto accade per ciò che viene costruito nel mondo fisico, le funzionalità e i servizi resi disponibili da un Digital Twin possono essere modificati e aumentare nel corso del tempo, per mezzo di nuove soluzioni software o grazie all'analisi dei *dataset* costituiti dai dati inviati dall'oggetto fisico. L'*augmentation* consente al Digital Twin di fornire funzionalità ulteriori rispetto a quelle realizzate "naturalmente" dall'entità fisica.

Ownership Questa caratteristica si riferisce al concetto di possesso del DT che deve essere regolato su due fronti diversi. Da un lato occorre stabilire chi sia il proprietario dell'enorme mole di dati generati e utilizzati dal Digital Twin stesso e dall'altro deve essere determinato il possessore del DT o, più nello specifico, del software che implementa la componente digitale del paradigma.

Nel caso di oggetti fisici, solitamente, è facile determinare chi ne sia il proprietario ma per le loro copie digitali può non essere così semplice e scontato perché spesso i due proprietari non coincidono. Il concetto di *ownership* risulta, dunque, essere molto complesso ma, data la diffusione dei Digital Twin, deve essere analizzato, gestito e regolato.

Servitization Il concetto di *Servitization* è sempre più centrale all'interno della trasformazione industriale denominata industria 4.0 e si riferisce alla vendita non solo del prodotto in sé ma anche dell'insieme dei servizi correlati o integrati al prodotto stesso, i quali diventano parte fondamentale del mercato.

I Digital Twin diventano un mezzo fondamentale per offrire un alto livello di *Servitization*, soprattutto grazie alla loro caratteristica di *augmentation* che amplia la quantità (e qualità) dei servizi che un prodotto può realizzare.

Previsione Dalla definizione del concetto riportata nella *Roadmap* della NASA si evince come la possibilità di utilizzare la copia virtuale di un oggetto

fisico per simularne il comportamento in ambienti differenti sia in realtà uno degli impieghi principali del paradigma. Questa applicazione dei Digital Twin li rende perfettamente integrabili con uno dei maggiori trend tecnologici degli ultimi anni, l'Intelligenza Artificiale. Ad esempio, l'integrazione delle due tecnologie abilita alla possibilità di realizzare simulazioni di tipo *what-if* [6].

Come detto in apertura, l'ampia diffusione del paradigma ha portato ad una mancanza di consenso generale circa la definizione e l'insieme delle caratteristiche essenziali dello stesso [16]. Le proprietà sopra riportate rappresentano per l'appunto il tentativo di Minerva *et al.* di fornire una visione consolidata del concetto.

Non tutti i requisiti precedentemente descritti sono essenziali per poter implementare un Digital Twin [1] ma l'insieme individuato fornisce sicuramente una visione generale e di valore del concetto.

1.3 Ambiti di applicazione del paradigma

Dalla sua nascita il concetto di Digital Twin trova naturale applicazione in due principali ambiti, quello manifatturiero e quello dell'aviazione. Più recentemente, però, il paradigma ha trovato diffusione in molteplici altri domini, tra i quali si trovano quello dell'*healthcare* e delle *Smart City*.

1.3.1 Industry 4.0

La formulazione del concetto di Digital Twin nasce proprio nel contesto manifatturiero e, in particolare, in relazione al Product Life Cycle Management. Quest'ultimo fa riferimento al processo che copre tutta la vita del prodotto, dalla sua ideazione e progettazione fino alla fase post-vendita.

In questo contesto, il paradigma dei Digital Twin può essere applicato o relativamente al singolo prodotto o in relazione all'intera fabbrica. Nel primo caso, i Digital Twin possono essere applicati per assistere la fase di progettazione del prodotto stesso e, successivamente, per monitorare la fase di produzione, al fine di individuare eventuali errori e discrepanze. Attraverso i DT è, dunque, possibile sincronizzare il design e la produzione del prodotto stesso [10]. L'impiego dei DT nella progettazione di un prodotto

abilita i progettisti stessi a realizzare valutazioni sulla qualità sin dalle prime fasi, consentendo, quindi, di realizzare il design in modo più efficiente ed informato. Relativamente al processo di produzione, i DT consentono di renderlo più affidabile, flessibile e predicibile [10]. Grazie allo scambio di dati tra il livello fisico e quello digitale, i DT permettono di monitorare complesse linee di produzione e di realizzare tempestivamente le attività di risoluzione dei problemi e di ottimizzare il processo.

Relativamente all'utilizzo dei Digital Twin per supportare l'intera fabbrica, i tre casi d'uso più frequenti sono: l'ottimizzazione dei processi, la manutenzione dei macchinari e la progettazione della loro collocazione ideale [6] [10] [20]. La possibilità di realizzare i DT dei diversi macchinari presenti all'interno della fabbrica, e di interconnetterli, consente di ottimizzare l'intero processo di produzione. Allo stesso tempo, i diversi DT permettono di monitorare lo stato dei singoli macchinari e di collezionare dati storici su cui impiegare modelli di Machine Learning e di AI al fine di prevedere eventuali malfunzionamenti. La progettazione della collocazione dei macchinari all'interno della fabbrica fisica può essere ottimizzata grazie all'impiego dei DT, i quali permettono di rendere modulare e parametrizzata l'architettura. In particolare, simulando la disposizione delle diverse componenti attraverso i DT è possibile testare, ottimizzare e validare le diverse soluzioni progettate e poi, dunque, realizzare quella che consente di accelerare la produzione e, contemporaneamente, ridurre i costi [6].

1.3.2 Healthcare

Oltre all'ambito manifatturiero e industriale, i Digital Twin stanno permeando velocemente, e in modo promettente, anche nel contesto sanitario, dove sono applicati per scopi differenti.

Uno dei primi impieghi del paradigma in ambito *healthcare*, riguarda la manutenzione predittiva e l'ottimizzazione delle performance dei dispositivi medici, sia in termini di consumo energetico che di velocità di esaminazione [6]. Un altro utilizzo frequente è, invece, quello volto alla pianificazione strategica dell'ospedale e dei sui processi [21], di cui un esempio concreto è rappresentato dai Digital Twin realizzati da General Electric Healthcare al fine di simulare e monitorare le operazioni ospedaliere [6] [22]. Un esempio ancora più rilevante riguarda la medicina personalizzata, la quale

richiede l'integrazione e l'elaborazione di un'ampia quantità di dati. Attraverso il paradigma è possibile generare una replica digitale dinamica dei singoli pazienti al fine di fornire ai medici grandi quantità di informazioni storiche [21]. Inoltre, la possibilità di realizzare un numero illimitato di copie digitali dei modelli delle reti dei fattori molecolari, fenotipici e ambientali rilevanti per i meccanismi della malattia nei singoli pazienti, abilita i medici a testare contemporaneamente molteplici trattamenti al fine di trovare quello più funzionale [23].

In relazione all'idea di generare pazienti digitali, molti degli studi in questo ambito sono volti a generare i DT del corpo umano o di alcuni dei suoi organi [24]. Siemens Healthineer, ad esempio, ha sviluppato un DT del cuore che simula i processi fisiologici sottostanti al fine di anticipare le risposte al trattamento prima dell'intervento concreto [3].

Un approccio alternativo a quelli descritti è rappresentato dall'idea di generare un ecosistema di DT interconnessi al fine di realizzare un sistema per il *Trauma Management* [3]. In questo caso, DT differenti, possibilmente prodotti da *vendor* diversi, sono sfruttati in modo sinergico al fine di supportare i medici nella gestione dell'attività di soccorso. L'idea su cui si basa è quella di generare un DT per ciascun *asset* fisico che risulti strategico all'interno dell'organizzazione sanitaria, al fine di riflettere e aumentare le funzionalità di ciascuno a livello digitale.

1.3.3 Smart City

L'impiego e il potenziale dei Digital Twin nel contesto delle Smart City incrementano di anno in anno grazie al crescente utilizzo di dispositivi IoT. Tra i molteplici possibili impieghi del paradigma nell'ambito delle Smart City risultano di rilievo quelli relativi alla gestione del traffico, del consumo energetico e degli edifici.

Attraverso l'utilizzo combinato dei Digital Twin e dei modelli di Machine Learning è possibile realizzare sistemi per monitorare e evitare la congestione del traffico, problema sempre più pressante a causa dell'importante incremento del numero di veicoli in circolazione. In particolare, questo è possibile grazie alla realizzazione dei DT dei mezzi in circolazione, dell'infrastruttura stradale e dei pedoni, i quali combinati all'utilizzo dei sistemi di

videosorveglianza consentono di modellare nel livello digitale il sistema del traffico.

Equipaggiare le città di molteplici sensori e tecnologie smart abilita alla possibilità di realizzarne i rispettivi Digital Twin. Quest'ultimi consentono, ad esempio, di realizzare simulazioni per rispondere a quesiti del tipo *what-if* e abilitano gli analisti a comprendere come le stesse città si comporterebbero in varie condizioni economiche, ambientali e sociali, e a identificare i possibili fattori di rottura [25].

1.4 Soluzioni tecnologiche disponibili

La diffusione del paradigma registrata negli ultimi anni ha consentito lo sviluppo di molteplici tecnologie che permettono di definire DT in modo più o meno strutturato e completo. Tra le principali soluzioni software che consentono di realizzare Digital Twins si trovano Azure Digital Twins di Microsoft, AWS IoT TwinMaker di Amazon Web Services e Eclipse Ditto, l'unica delle tre ad essere open source.

Azure Digital Twins Una piattaforma *as-a-service* che consente di creare grafi di DT. L'elemento caratterizzante di questa soluzione è il linguaggio che consente di definire i modelli dei DT: il Digital Twin Definition Language (DTDL). Il DTDL è un linguaggio JSON-like che permette di descrivere le diverse tipologie di entità, e quindi di DT, specificando l'insieme delle proprietà del loro stato, degli eventi di telemetria emessi, i comandi che sono in grado di processare e le relazioni che possono instaurare. Per mantenere aggiornati i singoli DT rispetto al *device* fisico, Azure Digital Twins può essere direttamente integrato con Azure IoT Hub, un nodo cloud in cui memorizzare tutti messaggi inviati da un dispositivo IoT. Alternativamente, al fine di integrare i DT con fonti di dati esterne, Azure mette a disposizione SDK per diversi linguaggi (C#, Java, Python, Go) e un API REST. Delle tre tecnologie citate, questa realizzata da Microsoft è sicuramene la più articolata e completa.

AWS IoT TwinMaker Un servizio cloud realizzato da Amazon Web Service che consente di definire Digital Twin principalmente in ambito indu-

striale. L'architettura attraverso la quale viene modellato il sistema fisico si basa, principalmente, sui concetti di "Entità" e "Componente'. Le entità sono le rappresentazioni digitali dei diversi elementi di un DT, ciascuna cattura una specifica capacità dell'elemento modellato. Esempi di elementi possono essere un'apparecchiatura fisica, un concetto o un processo. Alle entità sono associati dei componenti, i quali descrivono i dati e il contesto della relativa entità. I componenti possono fornire sia dati statici che flussi di dati originati attraverso ulteriori servizi AWS. Un tratto caratteristico di questa soluzione è l'importanza data all'aspetto di visualizzazione del DT. Infatti, vengono messi a disposizione gli strumenti per associare un modello 3D al Digital Twin definito.

Eclipse Ditto Un framework open source per costruire Digital Twin di *device* IoT. Nello specifico, Ditto è una piattaforma che consente di creare e gestire Digital Twin dando alle applicazioni la possibilità di astrarre dall'eterogeneità dei vari dispositivi fisici. Il concetto base utilizzato in Ditto è quello di *Thing*, il quale rappresenta un qualsiasi *asset* di un'applicazione IoT. Ogni *Thing* è identificata da un ID ed ha un insieme di attributi e di *feature*. Gli attributi consentono di descrivere più dettagliatamente la *Thing* e solitamente presentano le proprietà statiche. Le *feature*, invece, modellano ciascuna uno specifico insieme di dati e funzionalità associate alla *Thing*, la quale può averne un numero arbitrario. Similmente a Azure Digital Twins, ogni *Thing* è definita attraverso un file JSON che, rispetto ai modelli della soluzione di Microsoft, risulta, però, meno strutturato e semplice da definire. La peculiarità di questa tecnologia è data dalla presenza di un'API HTTP che consente di creare, gestire e utilizzare in modo completo le diverse *Thing*.

1.5 Problematiche

Nonostante il grande potenziale riconosciuto alla tecnologia dei Digital Twin e al forte interesse accademico e commerciale che vi gravita attorno, il paradigma si trova ancora in quella fase che può essere definita la sua infanzia [16]. Per questo motivo, attorno al concetto vi sono ancora delle problematiche, tecnologiche ma anche etiche e sociali, che devono essere affrontate e chiarite.

Le principali problematiche, sfide e dubbi che emergono in letteratura nei confronti dei Digital Twin rientrano in due macro-aree. La prima è quella relativa ai *dati* che vengono utilizzati, creati e scambiati dai DT, mentre la seconda riguarda la *standardizzazione* del concetto.

Di seguito, ciascuna delle problematiche riportate viene analizzata e discussa singolarmente più nel dettaglio.

1.5.1 Sicurezza, ownership e gestione dei dati

All'interno del paradigma dei Digital Twin i dati svolgono un ruolo cruciale, sono, infatti, uno dei fattori abilitanti. Un DT può adeguatamente modellare e diventare la copia virtuale di un oggetto fisico, unicamente se ha a disposizione una quantità sufficiente di dati per replicarne stato e comportamento. Conseguentemente, le principali preoccupazioni relative a questa tecnologia riguardano la privacy, la confidenzialità, la trasparenza e l'ownership dei dati [16].

Una delle componenti fondamentali di un DT è la connessione tra l'entità fisica e quella virtuale, attraverso questo legame, infatti, fluiscono tutte le informazioni che consentono alle due parti di avere una relazione simbiotica [26]. L'*entanglement*, riprendendo la definizione di questo componente data da Minerva *et al.* in [1], risulta essere, dunque, uno dei punti in cui è necessario garantire un impeccabile livello di sicurezza [26]. Il rischio di perdere informazioni o che queste siano corrotte o rubate durante il passaggio dal livello fisico a quello digitale può essere elevato. Per questo motivo, è necessario stabilire delle adeguate soluzioni di cyber-security, come, ad esempio, la crittografia dei dati stessi [26].

L'utilizzo dei DT nel contesto dell'IoT e in relazione con il *Cloud Computing* determina un altro motivo per il quale la progettazione dei Digital Twin stessi deve avvenire con una particolare attenzione alla robustezza nei confronti di attacchi hacker e alla penetrazione di virus [6].

A seconda dell'ambito di applicazione del paradigma, varia l'importanza di garantire riservatezza, anonimia dei dati e rispetto della privacy. Ad esempio, in ambito *healthcare* i dati trattati sono particolarmente sensibili, dunque, sono necessarie adeguate politiche relative alla privacy e alla sicurezza dei dati, in modo tale da poter contemporaneamente sfruttare i benefici

generati dall'adozione del paradigma e limitare eventuali inconvenienti e usi malevoli della tecnologia [1].

Lo stato di tecnologia emergente si nota anche dal fatto che la letteratura non ha trovato un accordo generale circa i requisiti minimi di sicurezza per il paradigma nonostante sia evidente la necessità di stabilirli.

Un altro aspetto problematico relativo ai dati, prodotti e utilizzati dai Digital Twin, riguarda l'*ownership* degli stessi [27]. *Chi possiede le informazioni sfruttate dal DT?*, *Chi possiede il DT stesso?*, quelle riportate sono le due principali domande che vengono poste relativamente alla problematica dell'ownership e per le quali mancano ancora delle risposte chiare ed applicabili. Stabilire l'ownership dei dati consente di determinare in modo chiaro chi può accedervi e in quale modo possono essere utilizzati [28].

Un ultimo aspetto critico riguardante i dati è quello relativo alla loro gestione. Il volume, la varietà e la velocità dei dati,che alcuni contesti di applicazione della tecnologia implicano, portano i Digital Twin a dover gestire quelli che sono a tutti gli effetti dei Big Data. Risulta quindi necessario considerare e affrontare anche le sfide relative all'utilizzo di questi dati, particolarmente complessi da gestire e analizzare [16].

1.5.2 Standardizzazione e interoperabilità

Una delle principali motivazioni che inibiscono l'adozione dei Digital Twin in molti settori e industrie è la mancanza di standard e di interoperabilità tra DT sviluppati con tecnologie e piattaforme differenti [26]. Questa complessità di integrazione delle diverse soluzioni software può essere eliminata solo attraverso un adeguato processo di standardizzazione.

Una delle principali iniziative di standardizzazione dei DT è rappresentata dal Digital Twin Consortium [29], il quale attraverso la collaborazione tra industrie, mondo accademico e entità governative, si dedica allo sviluppo globale del paradigma.

Un altro standard in fase di definizione è quello rappresentato dall'ISO 23247-1 [30], il quale ha l'obiettivo di definire un framework per i Digital Twin in ambito manifatturiero. Il framework specifica una serie di principi generali, un'architettura di riferimento e fornisce dei riferimenti per lo scambio di informazioni.

Come descritto nella sezione 1.2.1, due delle principali caratteristiche del paradigma sono la collaborazione e la compositività, le quali hanno un ruolo centrale nella possibilità di creare ecosistemi di Digital Twin. La cooperazione di più Digital Twin consentirebbe di sfruttare tutto il potenziale del paradigma stesso, ma tale possibilità viene ostacolata dall'assenza di Application Programming Interfaces (API) comuni [1]. Stabilire un insieme di API permetterebbe, allo stesso tempo, di continuare a sviluppare Digital Twin con diverse soluzioni software, e di avere comunque il vantaggio di avere un interfaccia comune che consenta l'interoperabilità [26].

Attualmente, l'utilizzo da parte delle industrie di interfacce e soluzioni software proprietarie creano quelli che vengono definiti, da Minerva *et al.* in [1], dei *silos* che di fatto impediscono la creazione di ecosistemi interoperabili.

Capitolo 2

Web of Digital Twins

In questo capitolo è descritto il concetto di *Web of Digital Twin* (WoDT) presentato in [2]. Inizialmente, viene fornita una panoramica del contesto in cui si colloca tale visione e delle motivazioni che hanno portato al suo concepimento.

Successivamente, è presentato il modello di Digital Twin adottato in WoDT, definendo le sue caratteristiche principali, il ciclo di vita e il sistema ad eventi su cui si basa.

In ultimo, sono delineati i servizi principali che una piattaforma software deve implementare e offrire per poter realizzare ecosistemi di Digital Twin.

2.1 Panoramica

L'idea di generare un Web of Digital Twins, ovvero un ecosistema dinamico, aperto e distribuito di Digital Twin interconnessi tra loro, nasce dalla volontà di modellare su larga scala realtà di entità fisiche tra loro interrelate.

Mentre le maggior parte delle visioni del paradigma Digital Twin riguardano la virtualizzazione di singoli *asset* fisici da utilizzare per applicazioni specifiche e verticali di un particolare ambito, la visione WoDT ha l'obiettivo di generare un approccio *DTs-as-a-service*, in cui le entità digitali non sono legate, esclusivamente, a specifiche applicazioni [3]. In altre parole, i Digital Twin diventano dei servizi Web che espongono delle API apposite, basate sugli standard e i protocolli tipici del Web e del Semantic Web, al fine di garantire interoperabilità a livello applicativo.

Come ampiamente discusso nel capitolo precedente, in letteratura non vi è ancora una visione allineata e univoca su come rappresentare i Digital Twin e come poterli sfruttare anche tra domini applicativi diversi. Le soluzioni software realizzate sono, infatti, principalmente ideate per uno specifico settore target e la visione del concetto di DT di ciascuna non risulta adeguatamente descritta, generando così una forte eterogeneità e frammentazione. Inoltre, in questo modo, si determina l'impossibilità di generare ecosistemi in cui *device*, servizi e utenti possano cooperare sfruttando a pieno le potenzialità del paradigma.

Per quanto riguarda il settore industriale, alcune attività di standardizzazione stanno emergendo. Ad esempio, l'*Industial Internet of Things Consortium* sta proponendo un'architettura di riferimento in cui vengono presi in considerazione concetti quali: relazioni tra DT, composizione e servizi di predizione, mantenimento e sicurezza. Purtroppo, anche in questo caso, la definizione del concetto è strettamente correlata al dominio specifico e il risultato è un ulteriore incremento della ramificazione di protocolli e formati dei dati a disposizione [31].

Il World Wide Web Consortium attraverso Web of Things (WoT) e l'organizzazione oneM2M sono anch'essi impegnati nel tentativo di fornire una descrizione e un accesso uniforme alle risorse fisiche per rendere l'IoT standard e interoperabile su più domini applicativi. In tale attività, però, la rappresentazione dei Digital Twin è ancora alle fasi iniziali, e vengono principalmente considerati come entità passive e marginali all'interno di architetture cloud, dove la definizione del loro comportamento è delegata a moduli software esterni, degradando di fatto il concetto di DT a una mera struttura dati per rappresentare risorse fisiche [31].

Tra le piattaforme software per realizzare DT, sviluppate dalle principali compagnie IT, Azure Digital Twin è quella che fornisce l'approccio più strutturato per progettare e sviluppare DT *cross-domain*, principalmente perché è dotata di un linguaggio chiamato Digital Twin Definition Language (DTDL) che consente di descrivere dei grafi di DT, rappresentando le loro proprietà e relazioni. Questa fondamentale *feature* è però limitata dalla scelta di utilizzare protocolli proprietari e dalla decisione di frammentare il processo di sincronizzazione dell'entità digitale rispetto a quella fisica su diversi servizi della piattaforma, invece di inglobarlo totalmente nel singolo Digital Twin.

L'utilizzo del paradigma che più si avvicina al concetto di WoDT è quello presentato in "The Gemini Principles" [32]. L'iniziativa prevede di connettere i diversi Digital Twin delle infrastrutture presenti sul territorio del Regno Unito, al fine di realizzare un National Digital Twin (NDT), una risorsa nazionale per migliorare la qualità e le prestazioni dei servizi coinvolti.

In questo contesto si colloca Web of Digital Twins, il quale fornisce una visione di ecosistema di DT ancora più ampia rispetto a quella del NDT, non limitando, infatti, la virtualizzazione solo a entità fisiche tangibili come edifici, ma ammettendo anche concetti più astratti come processi e attività. In letteratura, l'idea più affine a questa visione pervasiva della tecnologia è definita dal concetto di *Mirror Worlds*, presentato da D. Gelernter in [33]. Il principio è quello di definire dei modelli software di porzioni di realtà al fine di introdurre nuove funzionalità al di sopra dell'ambiente fisico, con l'obiettivo di supportare il lavoro individuale e di gruppo delle persone presenti nell'ambiente [3]. Web of Digital Twins può quindi essere considerato come un approccio concreto per progettare e sviluppare *Mirror Worlds* attraverso l'utilizzo del paradigma Digital Twin.

Web of Digital Twins è, quindi, definibile come un ecosistema aperto e dinamico di Digital Twin, rappresentabile attraverso un grafo, i cui nodi sono gli stessi DT e gli archi le relazioni che questi instaurano al fine di modellare anche i rapporti presenti nella realtà. In quest'ottica, i Digital Twin sono concepiti come entità software attive, dotate di un proprio ciclo di vita, capaci sia di riflettere lo stato della relativa entità fisica che di offrire funzionalità aggiuntive. Al fine di garantire interoperabilità a livello applicativo e di rendere i DT trasversali e non legati unicamente ad uno specifico settore, quest'ultimi espongo delle Application Programming Interfaces (API).

2.2 Il modello dei Digital Twin in WoDT

Le tre dimensioni su cui si sviluppa il paradigma, generalmente riconosciute in letteratura e riportate nel paragrafo 1.2, vengono in Web of Digital Twins ulteriormente definite e concretizzate. Per quanto riguarda l'elemento presente nel mondo reale, questo viene definito *physical asset* (PA) (risorsa fisica) con l'intenzione di potersi riferire ad una qualsiasi entità che ha una manifestazione o rilevanza nel mondo fisico e un tempo di vita ben definito.

La copia presente nel livello digitale è, invece, dotata di un modello specifico, il quale permette di catturare e rappresentare il PA ad un adeguato livello d'astrazione, cioè evitando aspetti non rilevanti per il suo scopo e modellando unicamente informazioni a livello di dominio e non tecnologiche. Infine, il legame tra la copia fisica e quella digitale viene definito *shadowing*, nello specifico, il termine definisce il processo che consente l'aggiornamento continuo e (quasi) in *real-time* dello stato interno del DT in relazione ai cambiamenti che avvengono nel PA.

All'interno di WoDT ciascun DT è, dunque, dotato di un modello (M) della relativa risorsa fisica, il quale definisce come il PA sia rappresentato nel livello digitale. La rappresentazione contenuta in M è definita in termini di:

- **Proprietà**: rappresentano gli attributi osservabili del relativo PA come dati etichettati i cui valori possono cambiare dinamicamente nel tempo, in modo conforme all'evoluzione dello stato del PA.

- **Eventi**: rappresentano gli eventi, a livello di dominio, che possono essere osservati nel PA.

- **Relazioni**: rappresentano i legami che sussistono tra il PA modellato e le altre risorse fisiche delle organizzazioni attraverso *link* ai corrispettivi Digital Twin. Anche le relazioni, come le proprietà, possono essere osservate, create dinamicamente e cambiare nel tempo, ma a differenza delle proprietà non sono propriamente elementi dello stato del PA.

- **Azioni**: rappresentano le azioni eventualmente invocabili sul PA attraverso l'interazione con il DT.

Definito il modello M, lo stato dinamico del DT (S_{DT}) può essere definito dalla seguente tupla:

$$S_{DT} =< P, E, R, A, t >$$

dove, P è l'insieme corrente delle proprietà e dei relativi valori, E è la sequenza degli eventi generati fino a quel momento, R è l'insieme corrente delle relazioni stabilite, A è l'insieme delle azione attualmente disponibili per poter essere invocate e t è il *timestamp* logico che rappresenta il tempo corrente del PA.

2.2.1 Il processo di shadowing

Il processo di *shadowing* consente di mantenere lo stato S_{DT} sincronizzato rispetto a quello della relativa risorsa fisica secondo quanto definito dal modello M. Nello specifico, ciascun aggiornamento rilevante dello stato del PA (S_{PA}) si traduce in una sequenza di 3 step principali:

1. ciascun cambiamento rilevante in S_{PA} viene modellato da un evento e_{PA};
2. l'evento viene propagato al DT;
3. dato il nuovo evento e_{PA}, lo stato S_{DT} viene aggiornato attraverso l'applicazione della funzione di *shadowing* ($Shad_{PA \to DT}$), la quale dipende strettamente dal modello M.

Secondo i tre step elencati, il nuovo stato aggiornato del DT (S'_{DT}) risulta quindi essere il risultato dell'applicazione della funzione di *shadowing*:

$$S'_{DT} = Shad_{PA \to DT}(S_{DT}, e_{PA})$$

Data la possibile complessità dell'entità fisica, il processo di *shadowing* deve essere in grado di considerare molteplici sorgenti di eventi e informazioni. Considerando, inoltre, la proprietà di composizione dei DT, le sorgenti di eventi che influiscono sul processo di aggiornamento del DT possono essere a loro volta dei DT.

Il processo di *shadowing* permette, inoltre, al DT di riflettere e invocare anche le possibili azioni del PA. Prendendo ad esempio il DT di una lampadina, questo dovrà fornire le azioni di accensione e spegnimento. Anche la propagazione delle azioni dal livello digitale a quello fisico avviene attraverso 3 step:

1. attraverso l'API esposta dal DT, viene richiesta un'azione a_{DT};
2. la richiesta di esecuzione dell'azione a_{PA} per il PA viene generata attraverso la funzione di *shadowing* ($Shad_{DT \to PA}$):

$$a_{PA} = Shad_{DT \to PA}(S_{DT}, a_{DT})$$

3. l'azione a_{PA} viene eseguita dal PA, causando un possibile cambiamento di stato della risorsa fisica stessa.

Un aspetto importate da sottolineare è che la richiesta di azione a_{DT} non cambia direttamente lo stato S_{DT}. Eventuali cambiamenti in S_{DT} possono, infatti, avvenire esclusivamente come risultato della funzione di *shadowing* dal PA al DT, come descritto precedentemente.

Una delle proprietà fondamentali del paradigma, generalmente riconosciuta in letteratura, è la capacità di *augmentation* del DT, cioè di estendere le funzionalità del PA virtualizzato, sfruttando appositamente il livello software in cui è realizzato [1]. Può essere considerata una specifica forma di *augmentation* anche la possibilità di realizzare predizioni e simulazioni attraverso il DT, considerando che tali funzionalità non sono presenti nella risorsa fisica.

In WoDT, la capacità di *augmentation* è modellata attraverso uno stato aumentato S_{AU}, che include un ulteriore insieme di proprietà, eventi, relazioni e azioni, e una funzione *Augm* che fa parte del modello M, insieme alla funzione *Shad*. Di conseguenza, quando si realizza un evento e_{PA}, oltre all'aggiornamento di S_{DT} attraverso la funzione di *shadowing* $Shad_{PA \to DT}$, anche lo stato S_{AU} viene modificato opportunamente secondo la funzione di *augmentation*:

$$S'_{AU} = Augm_{PA \to DT}(S_{AU}, S_{DT}, e_{PA})$$

Nel caso delle azioni a_{DT} esposte dal comportamento aumentato del DT, queste possono determinare, direttamente, un cambiamento nello stato S_{AU}, contrariamente a quelle derivanti dal *mirroring* della risorsa fisica, e possono causare la propagazione di un evento a_{PA} al PA, simmetricamente a quanto accade per il processo di *shadowing*:

$$a_{PA} = Augm_{DT \to PA}(S_{AU}, S_{DT}, a_{DT})$$

Lo stato aumentato S_{AU} e quello S_{DT} sono mantenuti separati al fine di rimarcare la differenza concettuale presente tra i due: le proprietà e le relazioni modellate dallo stato base del DT sono strettamente legate a quelle del PA e non possono essere direttamente modificate dal livello applicativo, al contrario delle componenti dello stato aumentato.

2.2.2 Modello di interazione

Il modello di interazione, che ciascun DT offre alle applicazioni presenti nel livello soprastante a quello di *Web of Digital Twins*, può essere suddiviso

in 3 funzionalità principali:

Invocazione di azioni Le applicazioni possono interagire con un DT al fine di invocare le azioni messe a disposizione dal suo modello. Seguendo il sistema ad eventi definito precedentemente, la richiesta di esecuzione di un'azione comporterà la generazione di un evento e_{DT} che sarà elaborato dalla funzione di *shadowing* per essere trasformato in un evento e_{PA}. La gestione di quest'ultimo da parte dell'*asset* fisico comporterà l'esecuzione effettiva dell'azione e potrà determinare una modifica nello stato dello stesso. Questo, conseguentemente, potrà determinare un cambiamento nello stato del Digital Twin. L'invocazione di un'azione dal livello applicativo, infatti, non può determinare direttamente una variazione nello stato del DT. In figura 2.1 sono sintetizzate le operazioni appena descritte.

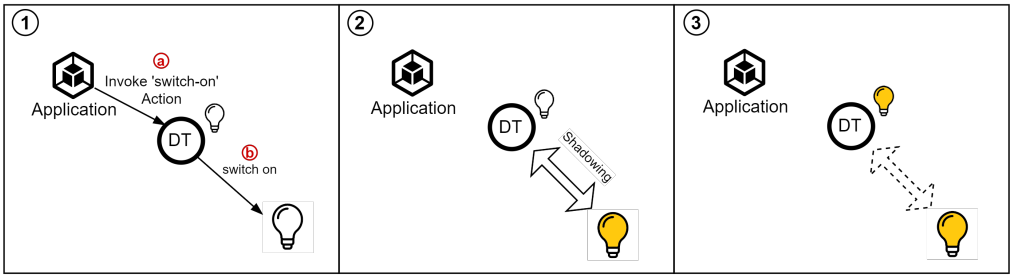


Figura 2.1: Invocazione di un'azione del DT da parte di un'applicazione

Osservazione Tipologia di interazione che abilita le applicazioni esterne, di qualsiasi tipo esse siano, al tracciamento delle modifiche nello stato di un DT. Le applicazioni possono ricevere gli eventi di interesse, previa sottoscrizione agli stessi. Tutte le proprietà, gli eventi e le relazioni presenti nel modello di un Digital Twin sono osservabili. Dunque, un applicativo che si sottoscrive, ad esempio, ad una proprietà riceverà una notifica ogniqualvolta il valore di quest'ultima sarà modificato. La registrazione alle varie componenti del modello può avvenire in modo dinamico, cioè un'applicazione può sottoscriversi, ad esempio, ad un evento, in qualsiasi momento nella vita del DT e allo stesso modo può cancellare la sua iscrizione quando lo desidera. Una rappresentazione grafica delle operazioni che consentono a un'applica-

zione di osservare una proprietà dello stato di un DT è fornita in figura 2.2.

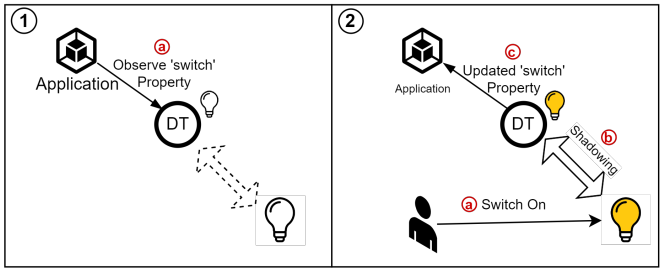


Figura 2.2: Osservazione dello stato del DT da parte di un'applicazione

Query Si fa riferimento alla possibilità di generare richieste *one-shot* relativamente allo stato del DT e al grafo di cui quest'ultimo fa parte. Tali richieste potranno essere semplicemente volte al recupero del valore di una proprietà oppure più articolate e formulate attraverso linguaggi per query come SPARQL. Questa tipologia di interazione consente anche di eseguire delle interrogazioni semantiche circa lo stato del Digital Twin. In figura 2.3 è schematizzata l'esecuzione di una query che consente all'applicazione di recuperare lo stato dell'entità fisica, attraverso il DT. Grazie al processo di *shadowing* il DT è sincronizzato rispetto allo stato attuale dell'entità fisica, dunque, quando l'applicazione richiede il valore di una proprietà può essere il DT stesso a fornirlo.

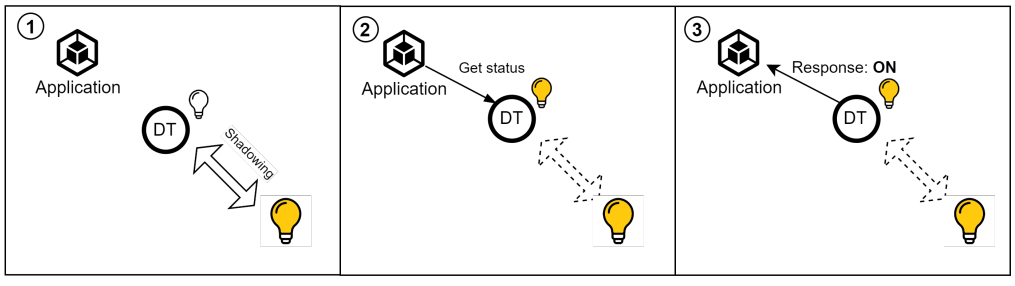


Figura 2.3: Query sullo stato del DT da parte di un'applicazione

2.2.3 Il ciclo di vita

La modellazione del concetto di DT fornita in Web of Digital Twins comprende anche la definizione teorica del suo ciclo di vita. Il *life cycle* definito prevede 5 stati attraverso i quali il DT transita dal momento in cui viene eseguito a quando viene dismesso. In figura 2.4 viene riportata una rappresentazione grafica del ciclo di vita, mentre, di seguito è fornita una descrizione dei suoi 5 stati:

1. *Operating & Not Bound*: è lo stato in cui il DT si trova a seguito della fase di inizializzazione, indica che tutti i moduli interni del DT sono attivi ma che non vi è ancora l'associazione con il relativo PA.

2. *Bound*: è lo stato in cui il DT transita a seguito della corretta esecuzione della procedura di *binding*. Quest'ultima consente di collegare le due parti e abilita al flusso bidirezionale di eventi.

3. *Shadowed*: è lo stato raggiunto dal DT quando inizia il processo di *shadowing* e il suo stato è correttamente sincronizzato con quello del PA.

4. *Out of Sync*: è lo stato che determina la presenza di errori nel processo di *shadowing*. Quando si trova in questo stato, il DT non è in grado di gestire né gli eventi di allineamento dello stato né quelli generati dal livello applicativo.

5. *Done*: è lo stato che il DT raggiunge quando viene fermato il processo di *shadowing* ma il DT continua ad essere attivo per poter gestire le richieste provenienti dalle applicazioni esterne.

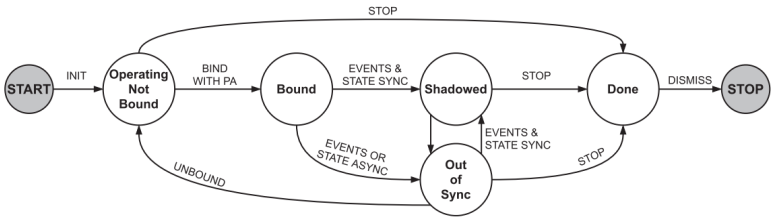


Figura 2.4: Rappresentazione degli stati e delle transizioni del ciclo di vita del DT. [2]

2.2.4 L'architettura

Nell'articolo di presentazione di Web of Digital Twins, oltre ad una descrizione teorica del concetto di Digital Twin, è stata delineata un'architettura ad eventi e modulare per sviluppare DT secondo il modello proposto. L'architettura presentata consente di avere flessibilità su tre aspetti fondamentali: il processo di *shadowing*, le funzionalità e il *deployment*.

La flessibilità nel processo di *shadowing* è data dalla possibilità di poter considerare PA eterogenei e adottare diversi modelli per eseguire il *mapping* tra il livello fisico a quello digitale, utilizzando i protocolli più funzionali.

La flessibilità a livello di funzionalità esposte dal Digital Twin stesso è uno di punti chiave di WoDT e l'architettura definita consente di implementare DT in ottica *as-a-service* che possono, quindi, essere adottati in sistemi aperti in cui sono coinvolte più organizzazioni.

Infine, per quanto riguarda la flessibilità a livello di *deployment*, l'architettura consente non solo un utilizzo in Cloud ma anche in Edge e Fog.

Una componente fondamentale dell'architettura proposta è il suo sistema di eventi. Nello specifico, ne vengono gestite tre tipologie:

- e_{PA}: sono gli eventi che rappresentano un cambiamento di stato nel PA, il quale può essere sia dovuto alla naturale evoluzione della risorsa che causato dal DT stesso a seguito della richiesta di un'azione da parte di un'applicazione.

- e_{DT}: sono gli eventi che rappresentano un cambiamento nello stato del DT, derivante dalla sua evoluzione in conseguenza al *mirroring* della risorsa fisica o generati dall'interazione con un'applicazione.

- e_I: sono gli eventi interni al DT, essenziali per il suo funzionamento. Questi eventi permettono di collegare la semantica del mondo fisico, rappresentata dagli eventi e_{PA}, a quella del mondo digitale, modellata dagli eventi e_{DT}, secondo il modello M definito dal progettista del DT.

Considerata la natura modulare dell'architettura, di seguito verranno presentati i moduli fondamentali, ognuno dei quali si basa un approccio *event-driven* e gestisce una o più delle tipologie di eventi appena descritte:

Event-driven Engine (EE) Il sistema nervoso del Digital Twin, in quanto tale, ha il compito di collegare tutte le componenti interne del DT e di abilitare la loro comunicazione reciproca attraverso gli eventi e_I.

Model Execution Engine (MEE) Il cervello del Digital Twin, in quanto tale governa tutte le altre componenti seguendo il modello (M) del DT definito dal progettista. Nello specifico, decide quali sono gli eventi che devono essere catturati e come questi influenzano lo stato e il comportamento del DT. Inoltre, definisce quali sono le possibili relazioni che il DT è in grado di gestire e quali sono le azioni che possono essere invocate.

Binding & Shadowing Module (BSM) Il cuore del Digital Twin, in quanto tale si occupa di realizzare l'operazione di *binding* del DT con il relativo PA e di gestire il processo di *shadowing*. Il processo di *binding* viene avviato una sola volta, tipicamente, alla creazione del DT e viene dismesso quando il PA viene rimosso. Questo modulo interagisce con l'Event-drive Engine per inviare gli eventi e_I al Model Execution Engine e riceve da quest'ultimo le politiche secondo le quali aggiornare lo stato del DT. In ultimo, il BSM si occupa di gestire il ciclo di vita del DT e permette la transizione nei diversi stati.

State Manager (SM) Il componente responsabile di gestire lo stato del Digital Twin, seguendo le regole del MEE, gli eventi notificati e il contesto di esecuzione.

Physical Asset Adapter (PAA) Il componente incaricato di catturare gli eventi e_{PA} generati dal PA associato al DT e di svolgere l'operazione duale, cioè di inviare gli eventi e_{PA} alla risorsa fisica, quando devono essere invocate delle azioni. Al fine di gestire la possibile eterogeneità di PA, in termini di protocolli e strutture dati utilizzati, il PAA deve mappare gli eventi che riceve nella rappresentazione uniforme e_I, in modo tale che possano essere gestiti dai moduli precedentemente descritti.

Digital Adapter (DA) Il componente complementare al PAA. Si occupa, infatti, di tradurre gli eventi e_I in e_{DT}, cioè gli eventi generati dal DT stesso.

Management Interface (MI) Il modulo che implementa tutte le funzionalità esposte dal DT unicamente agli altri DT, alla piattaforma WoDT e eventuali altre applicazioni di servizio.

I moduli descritti possono essere concettualmente classificati in due modi: elementi interni al singolo DT e elementi che consentono al DT di interfacciarsi con l'esterno. A questa seconda categoria appartengono il PAA, che consente al DT di relazionarsi con il proprio *asset* fisico, il DA, che permette la comunicazione tra il DT e le applicazioni e, infine, la MI, la quale permette l'interfacciamento con gli altri DT e la piattaforma.

2.3 Verso una piattaforma per WoDT

Per definizione *Web of Digital Twin* rappresenta un insieme aperto, dinamico e distribuito di Digital Twin che stabiliscono delle relazioni tra di essi, al fine di riflettere i rapporti presenti tra le relative entità nel mondo reale.

L'utilizzo dell'aggettivo "aperto" fa riferimento al fatto che i Digital Twin presenti nell'ecosistema non siano utilizzabili esclusivamente da un'unica organizzazione, cioè quella che li possiede, bensì possono essere sfruttati da più enti diversi per applicazioni differenti. Dunque, un'organizzazione può non solo sfruttare i propri DT ma anche quelli messi a disposizione dalle altre e nel medesimo ecosistema devono poter coesistere DT appartenenti ad organizzazioni distinte.

Per quanto riguarda la dinamicità dell'ecosistema, questa è data sia dal fatto che i Digital Twin possano essere aggiunti, e eventualmente rimossi, in momenti diversi, seguendo l'evoluzione del mondo fisico, sia dalla possibilità di creare e rimuovere le relazioni in modo dinamico al fine di riflettere le variazioni dei rapporti presenti nel mondo fisico.

Il termine "distribuito" indica, invece, che ciascun Digital Twin registrato possa essere in esecuzione in un nodo della rete differente da quello di tutti gli altri e che, quindi, ciascuno possa essere arbitrariamente eseguito in Edge come in Cloud.

Per poter realizzare un *web of DT* è necessario definire una piattaforma software in grado di fornire i servizi fondamentali per la creazione, la gestione e l'interrogazione dell'ecosistema stesso.

La necessità di generare un ecosistema dinamico determina che la piattaforma debba essere in grado di gestire in modo completo il ciclo di vita dei DT, cioè che debba assisterli dalla loro creazione alla dismissione. In particolare, l'inserimento di un Digital Twin nell'ecosistema si deve tradurre non solo nella sua creazione, esecuzione e *binding* con la relativa entità fisica ma anche nell'assegnazione di un identificatore univoco e di un indirizzo utilizzabile per interagire direttamente con il Digital Twin stesso.

Data la possibilità che organizzazioni diverse possano utilizzare il medesimo Digital Twin, anche senza esserne le proprietarie, è richiesto che sia implementato un servizio di ricerca dei Digital Twin, sia di tipo *white-page* che *yellow-page*. Un'applicazione esterna deve poter interrogare la piattaforma per ottenere l'indirizzo a cui contattare il DT desiderato, nello specifico, la richiesta può riguardare sia la traduzione di un identificatore nel suo corrispondente indirizzo sia la ricerca di un Digital Twin in base alle sue proprietà o caratteristiche.

WoDT può essere rappresentato attraverso un grafo orientato ed etichettato in cui i nodi sono i DT e gli archi le relazioni, dunque, la piattaforma deve consentire la navigazione del grafo creato, sfruttando le relazioni stabilite tra i Digital Twin e senza la necessità di avere a priori la conoscenza di quali siano tutti i DT presenti.

In ultimo, al fine di consentire interoperabilità e uniformità di accesso ai servizi, la piattaforma deve esporre delle API che le applicazioni esterne possono utilizzare per accedere all'ecosistema e alle varie funzionalità messe a disposizione.

Capitolo 3

Requisiti di una piattaforma per WoDT

L'obiettivo del seguente capitolo è quello di delineare l'insieme dei requisiti che devono essere soddisfatti per implementare un'infrastruttura software che consenta la realizzazione di ecosistemi di Digital Twin in ottica *Web of Digital Twins*. La piattaforma da realizzare, riprendendo la nomenclatura utilizzata in WoDT [2], è chiamata *WoDT Platform*.

Nel presentare i requisiti individuati si fa riferimento anche ad un semplice esempio, utile a chiarire e concretizzare alcuni aspetti cruciali. Il caso d'uso scelto è quello di una *Smart Home*, nell'edificio sono presenti due stanze: la cucina (*Kitchen*) e la camera da letto (*Bedroom*) e in ciascuna stanza sono presenti uno o più *smart device* che assolvono a compiti specifici, ad esempio nella cucina si trova la macchina del caffè (*Coffee Machine*).

3.1 Gestione di un ecosistema aperto e distribuito

Al fine di generare ecosistemi che rispettino i principi di *Web of Digital Twins* è necessario definire dei sistemi aperti e distribuiti, come analizzato nel paragrafo 2.3.

La necessità di definire un *open-system* implica il dover supportare, all'interno del medesimo ecosistema, Digital Twin appartenenti ad organizzazioni differenti, le quali, possibilmente, operano anche in domini distinti. L'utilizzo

del sistema da parte di molteplici organizzazioni, che hanno necessariamente obiettivi diversi, determina che la piattaforma e i DT stessi debbano poter essere sfruttati da applicazioni differenti.

Definizione dell'API dei Digital Twin La volontà di gestire un ecosistema aperto determina inoltre, l'esigenza di utilizzare standard e protocolli non proprietari al fine di garantire interoperabilità. Un altro requisito cruciale, che consente di rendere i DT fruibili in modalità *as-a-service* da applicazioni eterogenee, è quello di definire in modo chiaro ed omogeneo l'API di ciascuno. Quest'ultima, nello specifico, deve consentire di: recuperare le informazioni relative allo stato corrente del DT, invocare le azioni a disposizione, sottoscriversi agli eventi definiti dal modello del DT e navigare le relazioni stabilite dal DT all'interno dell'ecosistema. Al fine di non obbligare le applicazioni a conoscere a priori quali siano le possibili interazioni messe a disposizione da un DT, è necessario che sia il DT stesso a comunicare la propria API alle applicazioni che desiderano interagire con esso.

L'API che un Digital Twin espone deve consentire, inoltre, di comprendere a livello semantico cosa il DT rappresenti nel dominio in cui è collocato. Nello specifico, un'applicazione deve poter determinare di che cosa si occupi il DT, attraverso una visione ad alto livello dell'API. Per poter essere interoperabile in ottica *cross-domain*, ogni DT deve poter esporre la propria ontologia di riferimento, al fine di consentire alle applicazione di comprendere il significato dei termini utilizzati nell'API stessa. In questo modo, non deve necessariamente esistere, a livello di ecosistema, un'unica visione del mondo ma ogni DT può modellarlo come vuole, utilizzando i concetti a esso più congeniali senza impedire alle applicazioni di poter comunicare con DT diversi allo stesso tempo.

Uno degli aspetti che contraddistingue WoDT dalla maggior parte delle altre soluzioni che implementano il paradigma Digital Twin, è quello di considerare i DT come delle entità attive con uno specifico comportamento e, quindi, con una propria capacità di elaborare le informazioni provenienti dal mondo fisico. Questa caratterista rende i DT stessi dei servizi indipendenti che devono poter essere sfruttati dalle applicazioni in modo diretto senza la necessità che la piattaforma, attraverso la quale l'ecosistema si sviluppa, debba svolgere un ruolo da intermediario.

Supportare ecosistemi Edge-Cloud La possibilità che i Digital Twin possano appartenere ad organizzazioni differenti determina la necessità di gestire DT in esecuzione in punti diversi della rete. Inoltre, tale possibilità implica anche una forte eterogeneità a livello di *physical asset* modellati dai DT. Dunque, l'architettura della piattaforma deve essere in grado di supportare sia DT eseguiti in Cloud che in Edge. Tali requisiti determinano, quindi, che l'ecosistema abbia le caratteristiche di un sistema distribuito: ciascun DT può essere in esecuzione nel nodo della rete a esso più congeniale e la piattaforma è il componente del sistema che implementa i servizi di gestione dell'ecosistema stesso.

3.2 Gestione di un ecosistema dinamico

Oltre ad "aperto" e "distribuito", un altro aggettivo utilizzato per descrivere un ecosistema WoDT è "dinamico". L'obiettivo di modellare in modo pervasivo spaccati di realtà comporta, infatti, la necessità di gestire il dinamismo presente nel mondo reale. Alcune delle entità fisiche modellate possono far parte della propria organizzazione in modo stabile e continuativo per tutta la vita dell'organizzazione ma altre possono esistere e appartenere all'organizzazione solo per periodi limitati di tempo. I relativi Digital Twin devono, quindi, poter essere parte dell'ecosistema sin dal principio oppure essere aggiunti e rimossi in modo dinamico.

Creazione dinamica dei Digital Twin La piattaforma da sviluppare deve supportare la creazione e la disposizione dinamica dei Digital Twin. In particolare, la creazione può avvenire per mano dell'*owner* fisico del DT oppure attraverso il processo di *shadowing* di un altro DT già esistente. Nel primo caso, il soggetto fisico che possiede il DT deve poter richiedere alla piattaforma, attraverso una specifica API, l'inserimento nell'ecosistema del *Digital Twin* stesso. Nel secondo caso, invece, è un DT già presente nell'ecosistema che deve essere in grado di richiedere alla piattaforma la creazione di un nuovo DT. Verosimilmente, a seguito di tale richiesta il DT padre dovrà poter stabilire una relazione con il DT che ha richiesto di aggiungere.

Considerando l'esempio della Smart Home, il proprietario di casa deve poter richiedere alla piattaforma la creazione dei Digital Twin relativi alla casa

stessa e alle sue stanze mentre il DT del condizionatore (*Air Conditioner*) deve, ad esempio, poter autonomamente invocare l'inserimento del Digital Twin relativo ad un proprio processo di manutenzione. Quest'ultimo deve poter essere, inoltre, rimosso nel momento in cui la problematica segnalata viene risolta dall'intervento del manutentore.

La funzionalità di creazione dei Digital Twin offerta dalla piattaforma dovrà, quindi, essere sempre disponibile a prescindere dai DT presenti nell'ecosistema. Inoltre, dato il requisito di poter gestire un ecosistema distribuito e aperto, quando un DT viene aggiunto all'insieme, la piattaforma deve essere in grado di assegnargli un identificatore univoco e un indirizzo a cui contattarlo.

Gestione dinamica delle relazioni I rapporti che intercorrono tra le entità presenti nel mondo reale variano anch'essi nel tempo. Due *asset* fisici possono, ad un certo punto della loro vita, stabilire una relazione, la quale potrà o durare fino al termine del ciclo di vita di uno dei due o essere eliminata prima per altre ragioni. Ad esempio, nel contesto della *Smart Home* le relazioni tra i Digital Twin delle stanze *Kitchen* e *Bedroom* e quello della *Home* sono relazioni stabili che probabilmente sussisteranno per tutta la vita dei relativi DT mentre la relazione che si crea tra l'*Air Conditioner* e il DT del suo processo di manutenzione è una relazione che sarà creata nel momento in cui il DT del condizionatore necessita di manutenzione e sarà eliminata quando la relativa problematica sarà risolta.

La piattaforma da sviluppare deve, quindi, poter supportare anche la creazione e la disposizione dinamica delle relazioni tra i DT presenti nell'ecosistema, al fine di riflettere nella sua copia del grafo di *WoDT* le variazioni dei rapporti tra i DT. Nello specifico, la creazione di una relazione è un'operazione che il DT esegue in modo indipendente, ma quest'ultimo deve poter notificare alla piattaforma tale evento, in modo tale che la *WoDT Platform* possa mantenere aggiornato il grafo rappresentante l'ecosistema.

Evoluzione dei DT Un altro aspetto da considerare, correlato al dinamismo presente nel mondo reale, è quello relativo al cambiamento delle proprietà e delle funzionalità esposte dagli *asset* fisici. Un cambiamento nel modello del PA associato ad un DT può determinare una variazione nel mo-

dello M del Digital Twin stesso. Ad esempio, la variazione delle azioni messe a disposizione dall'entità fisica può comportare la modifica dell'API esposta dal Digital Twin alle applicazioni esterne. La piattaforma, al fine di poter offrire correttamente alcune delle sue funzionalità, deve poter essere notificata dai DT quando si realizzano variazioni alle loro API.

3.3 Interazione con un Digital Twin

Un requisito fondamentale che deve essere soddisfatto dalla *WoDT Platform* è quello di consentire alle applicazioni esterne di interagire con ciascuno dei Digital Twin presenti nell'ecosistema. Come descritto nel capitolo precedente, le 3 tipologie di interazioni messe a disposizione dai singoli Digital Twin sono: l'invocazione di un'azione, l'osservazione dello stato del DT e l'esecuzione di query. Il compito della piattaforma, in questi casi, deve essere unicamente quello di consentire alle applicazioni esterne di *scoprire* a quale Digital Twin rivolgersi. In quanto entità attive, infatti, ciascun DT deve essere in grado di supportare tali interazioni in modo autonomo.

Osservazione di un Digital Twin Seguendo il modello di Digital Twin proposto in WoDT, la possibilità di osservare le proprietà, gli eventi e le relazioni di un DT deve essere fornita alle applicazioni mediante uno specifico *Digital Adapter* dello stesso DT. Le applicazioni devono potersi registrare presso una o più componenti dello stato del DT come osservatori, al fine di ricevere degli eventi quando tali componenti vengono modificate. Le modalità con cui le applicazioni possono sottoscriversi e cancellare la loro iscrizione al flusso di eventi devono essere specificate dall'API esposta da ciascun DT.

Riepilogando, un DT deve consentire alle applicazioni la possibilità di registrarsi per ricevere gli eventi relativi alle variazioni di stato, cioè deve abilitare l'osservazione asincrona delle proprie componenti, eventualmente anche quelle dello stato aumentato. Ogni proprietà, relazione e evento deve poter essere osservato in modo indipendente e senza determinare interruzioni nell'esecuzione del processo di *shadowing* del DT stesso.

Riprendendo l'esempio della *Smart Home*, l'applicazione mobile che consente di monitorare e gestire la casa deve poter osservare la proprietà del DT *Air Conditioner* relativa alla temperatura a cui è impostato il condizionatore,

in modo tale da mostrare al proprietario di casa il valore corrente aggiornato. Il sistema di monitoraggio dell'azienda che si occupa della manutenzione del condizionatore deve, invece, potersi sottoscrivere all'evento che segnala eventuali problematiche, al fine di notificare tempestivamente gli addetti quando il condizionatore fisico necessita di un intervento di manutenzione.

Il requisito di realizzare dei DT fruibili in modalità *as-a-service* determina la necessità che ciascun DT possa offrire la funzionalità di osservazione in modo indipendente dalla *WoDT Platform*. Quest'ultima però, considerando la possibilità che l'ecosistema sia utilizzato da molteplici organizzazioni, deve consentire alle applicazioni di determinare a quale DT rivolgersi, cioè deve offrire un servizio di ricerca. Tale funzionalità verrà analizzata nel paragrafo 3.4.

Query su un singolo Digital Twin L'altra tipologia di interazione fondamentale che ciascun DT deve fornire, in maniera indipendente, è quella relativa alle *query*. Un'applicazione deve poter realizzare delle interrogazioni sullo stato di un DT, in particolare, deve poter sia indagare il valore corrente di una delle proprietà, sia formulare delle richieste relative al sotto-grafo definito dalle relazioni stabilite dal DT. Anche questa funzionalità deve essere realizzata mediante uno specifico *Digital Adapter*.

Le interrogazioni che un'applicazione può realizzare devono poter essere arbitrariamente complesse, dunque, sarà necessario supportare anche eventuali linguaggi di query semantiche.

Come per la funzionalità precedentemente presentata, anche la possibilità di interrogare un DT non richiede alcun ruolo da intermediario per la *WoDT Platform*. Occorre però specificare che entrambe le funzionalità non devono interferire o bloccare il processo di *shadowing* dei DT. Dunque, gli aggiornamenti provenienti dal PA hanno la precedenza rispetto alla gestione di queste interazioni.

Relativamente al contesto della *Smart Home*, un esempio di interrogazioni possibili sono: individuare tutte le stanze che hanno un condizionatore oppure, determinare qual è la temperatura registrata all'interno della camera da letto. La prima query potrebbe essere risolta dal Digital Twin *Home* mentre la seconda potrebbe essere soddisfatta dal DT *Bedroom*, nel caso in cui esso fosse già dotato della proprietà *temperature*, oppure potrebbe essere

in qualche modo demandata al DT *Air Conditioner* con cui il precedente DT ha una relazione.

3.4 Interazione con l'ecosistema

Le interazioni di *osservazione* e *query* possono interessare non solo il singolo DT ma anche l'intero ecosistema. Un'altra tipologia di interazione che coinvolge l'intero insieme dei DT e che deve essere gestita a livello di piattaforma è la ricerca dei Digital Twin presenti nell'ecosistema. Di seguito vengono analizzate ciascuna delle interazioni citate al fine di determinare quali siano i requisiti da soddisfare per implementarle adeguatamente.

Discovery dei Digital Twin Il servizio di *discovery* dei Digital Twin è una delle funzionalità fondamentali che la *WoDT Platform* deve implementare. Tale servizio consiste nel consentire alle applicazioni esterne di determinare l'indirizzo associato a ciascun Digital Twin, in modo tale da poter poi richiedere autonomamente a ciascun DT la propria API.

La piattaforma deve consentire sia una ricerca dei DT attraverso il meccanismo delle pagine bianche (*white pages*) sia attraverso quello delle pagine gialle (*yellow pages*). Realizzare un servizio di *white pages* significa consentire alle applicazioni esterne di determinare l'indirizzo a cui contattare un DT conoscendo il suo identificatore. La modalità *yellow pages* implica, invece, la possibilità di determinare quali sono i Digital Twin che soddisfano determinate proprietà. In particolare, per implementare quest'ultima funzionalità è necessario che quando un DT viene registrato nell'ecosistema siano forniti dei metadati necessari per classificarlo.

Considerando l'esempio della *Smart Home*, l'applicazione utilizzata dal manutentore dei condizionatori potrebbe richiedere alla piattaforma di fornirgli, attraverso il suo servizio di pagine gialle, tutti gli indirizzi dei Digital Twin classificati come *device* o *air conditioner*. In questo modo, l'applicazione potrà automaticamente rivolgersi a ciascuno per individuare le diverse problematiche.

Osservazione dell'ecosistema Il tema dell'osservazione dell'ecosistema *WoDT* è ancora oggi un tema aperto e non del tutto esplorato e modellato

chiaramente.

In generale, osservare l'intero ecosistema o un suo sotto-grafo significa consentire ad un'applicazione di definire quali tipologie di eventi ricevere relativamente ad un insieme di DT. Ad esempio, nel contesto della *Smart Home*, un'applicazione potrebbe voler osservare tutte le stanze in modo tale da essere notificata quando in una o più di queste viene accesa la luce. Tale richiesta potrebbe essere formulata al DT *Home* se si vuole considerare la totalità delle stanze presenti nella casa, mentre, se si volesse considerare solo le stanze di un determinato piano, la sottoscrizione dovrebbe avvenire nei confronti del DT relativo al piano di interesse, il quale considererà solo il suo sotto-grafo di stanze.

Le modalità con cui esprimere la tipologia di evento da osservare e dunque, l'insieme di DT da considerare, non sono ancora definite in modo chiaro. Nel contesto di questa tesi dunque, questo requisito non è stato del tutto progettato e implementato.

L'osservazione di un ecosistema si può tradurre, inoltre, nella possibilità di essere notificati nel momento in cui viene aggiunto o rimosso un Digital Twin oppure quando vengono create o eliminate delle istanze delle relazioni. Tale funzionalità deve consentire alle applicazioni di monitorare in modo asincrono lo stato generale dell'ecosistema, senza interferire con gli altri servizi della *WoDT Platform* o con le attività dei singoli Digital Twin. In particolare, proprio come nel caso dei singoli DT, deve essere possibile osservare singolarmente ciascun aspetto e le applicazioni possono decidere, in qualsiasi momento, di terminare la loro sottoscrizione agli eventi per i quali si erano registrate.

Ad esempio, l'applicazione mobile che gestisce la *Smart Home* vuole poter osservare l'intero ecosistema al fine di essere notificata quando un nuovo *device* viene aggiunto o quando l'*Air Conditioner* stabilisce una relazione con il DT del proprio processo di manutenzione, in modo tale da poter riflettere lo stato generale della casa.

Query sull'ecosistema La possibilità di realizzare query che coinvolgono i Digital Twin dell'intero ecosistema o anche solo una sua porzione è, come il precedente, un tema non del tutto esplorato. Similmente al requisito dell'osservazione dell'ecosistema, questa funzionalità potrebbe essere affidata al

DT radice dell'intero grafo o del sotto-grafo di interesse, oppure, potrebbe essere una funzionalità svolta a livello di piattaforma.

Tali query possono riguardare richieste come: la lista di tutti i Digital Twin presenti nell'ecosistema che rispettano determinate caratteristiche, il grafo con le relazioni attualmente presenti tra i diversi DT e interrogazioni più articolate formulate con linguaggi di query specifici. In modo speculare a quanto richiesto per le altre tipologie di interazioni, anche per quanto riguarda le query sull'ecosistema, queste non devono interferire con il corretto funzionamento della *WoDT Platform* o con quello dei DT.

Nel contesto della *Smart Home*, esempi di query che un'applicazione potrebbe generare sono: il numero di *device* con determinate caratteristiche presenti nella casa, le diverse temperature registrate nelle varie stanze in cui è presente un condizionatore acceso e così via.

Capitolo 4

Progettazione della WoDT Platform

In questo capitolo viene presentata la fase di progettazione della *WoDT Platform*, avvenuta in seguito all'analisi dei requisiti descritta nel precedente capitolo. L'obiettivo della fase di progettazione è stato quello di determinare quale fosse l'architettura necessaria per implementare le diverse funzionalità emerse dall'analisi. Nello specifico, si sono individuati l'insieme dei moduli che costituiscono l'intera architettura della *WoDT Platform* e quelli da aggiungere all'architettura del singolo Digital Twin presentata nella sezione 2.2.4.

Il capitolo presenta inizialmente l'architettura finale della piattaforma e dei Digital Twin, analizzando in dettaglio ciascuna delle componenti individuate. In seguito, sono presentati i modelli di interazione tra le applicazioni esterne, la piattaforma e i DT. Anche in questo caso, dove necessario, viene utilizzato l'esempio della *Smart Home* per chiarire gli aspetti più complessi.

4.1 Architettura della WoDT Platform

La progettazione dell'architettura della *WoDT Platform* è stata fortemente influenzata dalla necessità di poter gestire degli ecosistemi distribuiti. Al fine di soddisfare questo requisito, infatti, si è deciso di modellare il concetto di nodo della piattaforma, chiamato appunto *WoDT Platform Node*.

Si ha l'esigenza di definire un sistema distribuito in quanto non solo i DT possono appartenere ad organizzazioni differenti e quindi, devono poter essere in esecuzione in punti diversi della rete, ma anche perché all'interno della stessa organizzazione può esservi l'esigenza di eseguire i DT in determinati nodi computazionali. Ad esempio, potrebbero sussistere requisiti di performance o di hardware specifici su cui implementare il DT. Nel caso in cui si avesse la necessità di una comunicazione *real-time* tra il DT e la relativa risorsa fisica, ad esempio, una possibile soluzione sarebbe quella di eseguire il DT in Edge, cioè su un nodo quanto più possibile vicino all'entità fisica.

Il *WoDT Platform Node* è stato quindi progettato in modo tale da poter supportare l'esecuzione dei Digital Twin anche quando questi non sono nel medesimo punto della rete in cui è stato avviato il nodo della piattaforma. In figura 4.1, è riportato un esempio di come un ecosistema WoDT può articolarsi rispetto ai nodi di esecuzione presenti nella rete. Il caso "Edge A" può essere definito come il caso concentrato, cioè quello in cui i DT e il nodo della piattaforma sono in esecuzione sul medesimo server, mentre nel caso "Edge B" è possibile osservare come due DT siano in esecuzione su un nodo in cui non è presente il *layer* della piattaforma, questo perchè sono gestiti dal *WoDT Platform Node* in esecuzione sul *Master execution Node*. Infine, nel caso "Cloud" si è modellata la possibilità che i DT e il nodo della piattaforma siano eseguiti totalmente in cloud.

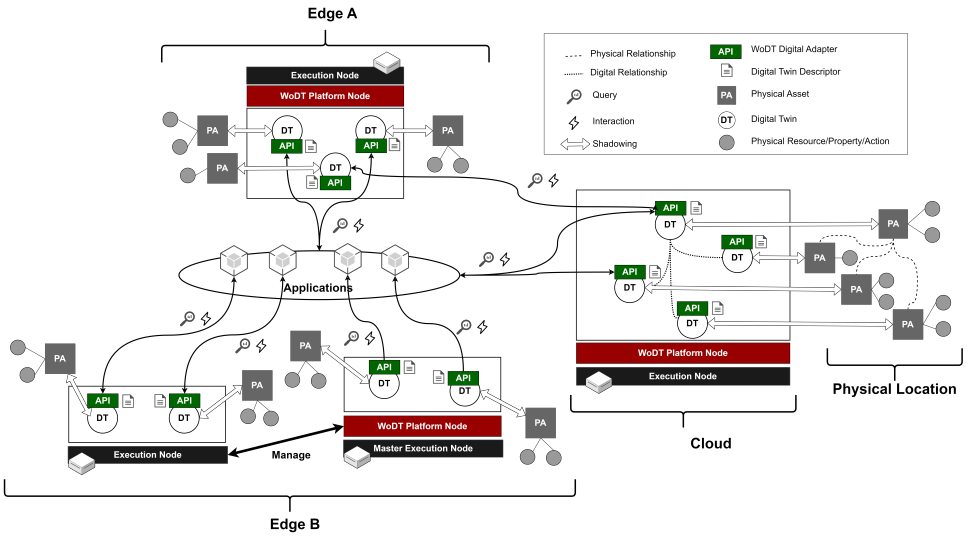


Figura 4.1: Rappresentazione di un ecosistema WoDT

Il *WoDT Platform Node* presenta un'architettura modulare al fine di decomporre la soluzione del problema, rendere l'implementazione più semplice e ottenere un'architettura scalabile e più facile da mantenere e estendere. In figura 4.2 viene riportata una rappresentazione dei moduli che compongono l'architettura di un *WoDT Platform Node* e di seguito viene descritta ciascuna componente.

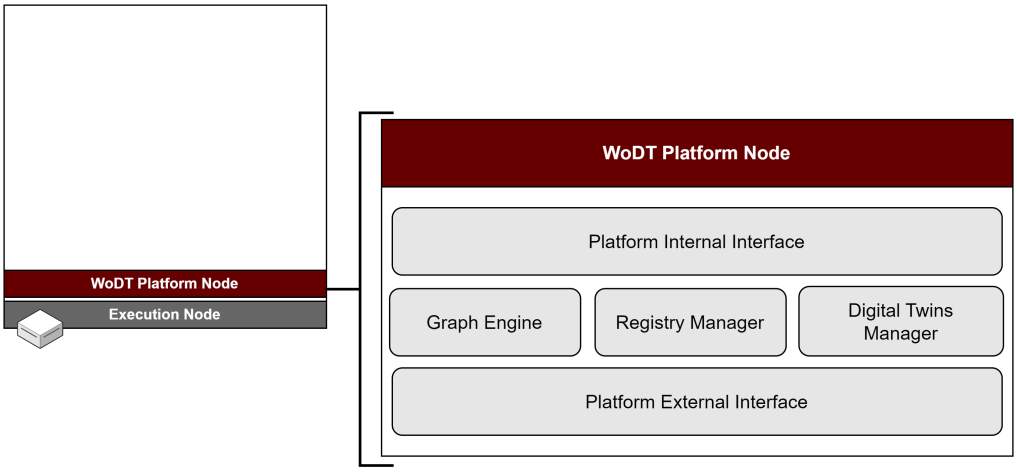


Figura 4.2: Architettura interna del WoDT Platform Node

Platform External Interface L'interfaccia di comunicazione esterna della piattaforma responsabile di gestire le interazioni con il livello applicativo. Questo componente riceve tutte le richieste provenienti dall'esterno dell'ecosistema come, ad esempio, quelle di creazione e *discovery* dei Digital Twin. Volendo realizzare una soluzione basata sui protocolli propri del Web, questa interfaccia è stata progettata come un server HTTP REST. Il modulo si occupa di direzionare le richieste ai componenti interni in grado di soddisfarle, di fatto rappresenta il punto di ingresso del nodo della piattaforma per tutte le entità esterne all'ecosistema.

Platform Internal Interface L'interfaccia di comunicazione interna della piattaforma, gestisce le interazioni con i Digital Twin presenti nell'ecosistema. Attraverso questo modulo, i Digital Twin in esecuzione possono richiedere la creazione di altri DT e la traduzione degli identificatori in indirizzi,

inoltre, possono notificare il loro corretto inserimento nell'ecosistema e la creazione o eliminazione di una relazione.

Anche questo modulo non si occupa direttamente di soddisfare le richieste bensì le ridireziona verso i componenti interni del nodo in grado di elaborarle. La *Platform Internal Interface* è, quindi, il punto di ingresso della piattaforma per tutte le entità interne all'ecosistema.

Si è scelto di progettare anche questo componente come un servizio REST al fine di mantenere omogeneità nei protocolli utilizzati per realizzare la piattaforma.

Digital Twins Manager Modulo interno incaricato di gestire il ciclo di vita dei Digital Twin, dalla loro creazione alla dismissione. Nello specifico, questo componente si occupa di gestire le richieste di inserimento dei DT e di eseguirli, assegnando loro un identificativo univoco e un indirizzo.

Il *Digital Twins Manager* si sottoscrive alla *Platform External Interface* per soddisfare le richieste di creazione e rimozione dei Digital Twin mentre si interfaccia con la *Platform Internal Interface* per essere notificato dai Digital Twin quando la loro creazione è andata a buon fine, per ricevere eventuali aggiornamenti sulla loro esecuzione e per eseguire le richieste di creazione provenienti dai DT stessi.

Questo modulo si relaziona anche con gli altri componenti interni al nodo, al fine di notificarli quando un nuovo Digital Twin è stato inserito o quando un DT già presente viene rimosso.

Registry Manager Il registro che mantiene l'elenco dei Digital Twin inseriti nell'ecosistema e memorizza per ciascuno gli eventuali *metadati* che consentono la classificazione. Questo componente si occupa di fornire il servizio di *discovery* dei Digital Twin sia per le applicazioni esterne che per i DT presenti nell'ecosistema, per questo motivo si sottoscrive ad entrambe le interfacce di comunicazione della piattaforma.

Il *Registry Manager* viene, inoltre, notificato dal *Digital Twins Manager* in caso di creazione o dismissione di un DT, al fine di poter aggiornare adeguatamente il proprio elenco.

Graph Engine Componente incaricato di gestire il grafo dei DT registrati e che consente la navigazione dell'interno ecosistema. Tiene traccia sia della registrazione dei DT, comunicando con il *Digital Twins Manager*, sia della creazione e rimozione delle relazioni, sottoscrivendosi all'*Internal Platform Interface*. Relativamente all'interazione con il livello applicativo, il *Graph Engine* è il componente adibito alla risoluzione delle query riferite all'intero ecosistema.

4.2 Architettura dei Digital Twin

La progettazione dell'architettura dei Digital Twin ha consistito principalmente nell'individuare come strutturare alcune delle componenti del modello proposto in *Web of Digital Twins* e riportato nel paragrafo 2.2, al fine di realizzare i requisiti individuati. In particolare, sono stati progettati il WoDT Digital Adapter e la Management Interface.

Il requisito di generare e gestire un ecosistema aperto ha determinato la necessità di progettare adeguatamente l'API dei Digital Twin, i quali, in quanto entità attive, sono in grado essi stessi di fornire al livello applicativo alcuni servizi. Nello specifico, si è progettato quello che è stato denominato *Digital Twin Descriptor* (DT-Descriptor), cioè un documento che consente alle applicazioni di determinare come interagire con il DT. Questo fondamentale elemento verrà presentato nella sezione 4.2.1.

In figura 4.3 vengono rappresentati i moduli principali che compongono l'architettura interna di un Digital Twin; di seguito vengono presentati i due componenti che sono stati oggetto della fase di progettazione di questo progetto di tesi.

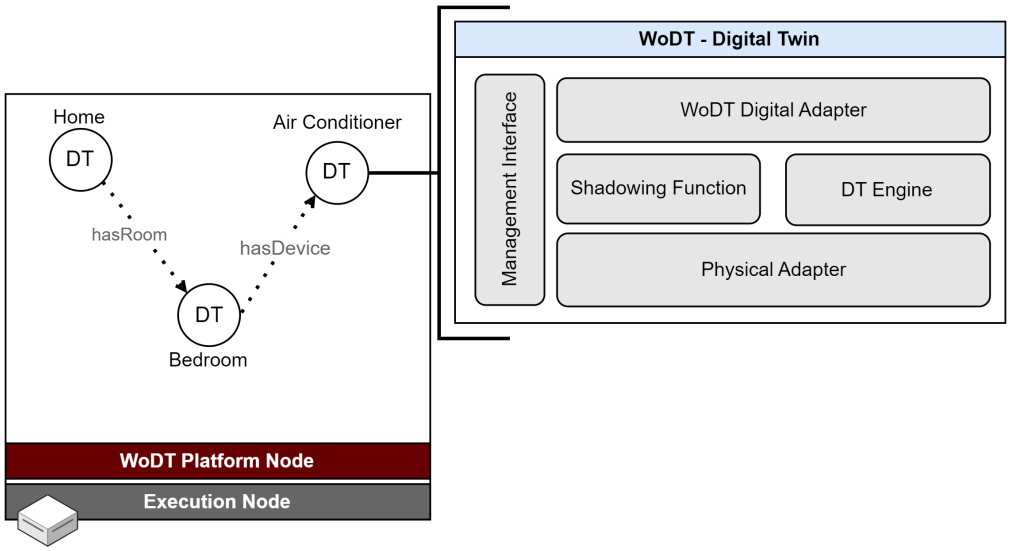


Figura 4.3: Rappresentazione dell'architettura interna di un Digital Twin

Management Interface Interfaccia che consente al Digital Twin di relazionarsi con la *WoDT Platform*. Nello specifico, consente al DT di notificare alla piattaforma il suo corretto inserimento nell'ecosistema, la creazione e la rimozione delle istanze delle relazioni.

Attraverso questo componente, inoltre, il modulo di *shadowing* può richiedere alla piattaforma la traduzione degli identificatori dei DT nei corrispondenti indirizzi o l'inserimento di un nuovo Digital Twin. La prima funzionalità è particolarmente utile quando il DT vuole stabilire una relazione con un DT di cui conosce solo l'id.

Di fatto, la progettazione della *Management Interface* è stata influenzata anche dalla decisione di strutturare la *Platform Internal Interface* come un web server REST. Infatti, conseguentemente a tale decisione, il modulo è stato progettato come un *client* HTTP che viene utilizzato dai moduli interni del DT per sfruttare i servizi esposti dall'API della *Platform Internal Interface*.

WoDT Digital Adapter Digital Adapter (DA) progettato appositamente per integrare il Digital Twin al *Web of Digital Twin*. In quanto DA ha l'obiettivo di interfacciare il DT con il livello applicativo, offrendo alle applicazioni un'interazione di tipo *as-a-service* con il Digital Twin stesso.

Nello specifico, il *WoDT Digital Adapter* consente al Digital Twin di esporre e implementare il proprio DT-Descriptor, secondo il requisito di generare un ecosistema aperto. Come conseguenza di quest'ultimo, inoltre, si è deciso di progettare tale DA come un server HTTP, al fine di adottare gli standard web e permettere la libera fruizione del DT.

Grazie a questo componente, rispetto al livello applicativo, i Digital Twin diventano un servizio vero e proprio attraverso il quale indagare lo stato dell'entità reale, interagire con essa mediante le azioni esposte e visualizzare informazioni aggiuntive grazie alla capacità di *augmentation* del DT stesso.

4.2.1 Digital Twin Descriptor

Con l'obiettivo di generare un ecosistema aperto e rendere i Digital Twin delle entità attive e interoperabili, ciascun DT espone la propria API alle applicazioni presenti sopra al livello del *Web of Digital Twins*. Nello specifico, l'API consente sia di recuperare le informazioni relative allo stato dell'entità sia di interagire con essa attraverso le azioni a disposizione.

Come spiegato in precedenza, grazie al *WoDT Digital Adapter* i DT diventano dei servizi web REST con una specifica API che viene descritta da un apposito documento esposto sulla rotta di default del DT stesso. Il documento che sintetizza l'API viene chiamato *Digital Twin Descriptor* (DT-Descriptor) e presenta una struttura standard che consente di specificare quali siano le componenti del modello del DT di riferimento e come è possibile interagire e sfruttare ciascuna di esse.

Il DT-Descriptor può essere considerato il punto di ingresso del Digital Twin e fornisce l'insieme delle possibili interazioni al fine di integrare i diversi Digital Twins e soprattutto le applicazioni esterne. Per questo motivo, il DT è responsabile sia di esporlo sia di mantenerlo aggiornato rispetto alle variazioni del proprio modello, in modo tale da garantire il suo corretto utilizzo.

La progettazione del *Digital Twin Descriptor* è stata ispirata dal concetto di *Thing Description*(TD) proposto in *Web of Things* e dal *Digital Twin Definition Language*(DTDL) formulato da Microsoft. La TD è una specifica, standardizzata dal W3C, che consente di descrivere i *device* IoT e l'insieme delle *affordance* che possono essere utilizzate per interagire con essi. Il

DTDL è un linguaggio che consente di definire il modello dei Digital Twin implementati attraverso la piattaforma cloud: *Azure Digital Twin*.

Il descrittore realizzato non ha l'obiettivo di essere un'alternativa alle due soluzioni citate bensì si pone come una rappresentazione formale che consente di descrivere gli elementi necessari a supportare la visione di un *Web of Digital Twins* aperto e interoperabile. Nel presentare il *Digital Twin Descriptor* saranno evidenziati sia gli aspetti comuni alle due tecnologie sia, in particolar modo, le sue peculiarità. Queste ultime, infatti, determinano le motivazioni per le quali non è stato possibile adottare direttamente nessuna delle due specifiche già affermate.

Entrambe le fonti di ispirazione codificano la loro specifica in formato JSON-LD e dunque adottando una sintassi JSON. Anche il *Digital Twin Descriptor* è stato progettato adottando tale formato, in quanto si pone come lo standard per lo scambio di dati sul web. JSON-LD non solo risulta semplice da leggere e scrivere per un umano ma consente anche di essere processato facilmente dalle applicazioni. Proprio quest'ultimo aspetto risulta fondamentale, in quanto il fatto che il DT-Descriptor sia *"machine-interpretable"* consente l'interazione dinamica tra le applicazioni e il DT stesso, anche quando queste non hanno a priori conoscenza del dominio modellato dal DT.

Una delle differenze sostanziali tra il DT-Descriptor e il DTDL è data dal fatto che il descrittore permette di definire formalmente le modalità di interazione di un Digital Twin specifico, cioè come è possibile osservare le proprietà e gli eventi, invocare le azioni e indagare le relazioni stabilite dallo stesso, mentre il linguaggio di Microsoft consente di definire un *modello* che diverse istanze di DT posso adottare. Sostanzialmente, quello che il DTDL consente di esplicitare è paragonabile al concetto di *classe* presente nella programmazione ad oggetti. Da questo punto di vista, il DT-Descriptor è molto più simile al concetto di *Thing Description*, il quale, per l'appunto, modella le *affordances* di un'istanza specifica di device IoT.

Un aspetto presente in tutte e 3 le specifiche, in quanto ereditato dal formato JSON-LD, è una prima parte del documento stesso in cui vengono presentati una serie di metadati riferiti all'entità. Il descrittore, in seguito, presenta per ciascuno degli elementi del modello del Digital Twin (proprietà, eventi, azioni e relazioni) le possibili tipologie di interazioni, similmente a quanto avviene nella TD. Nel listato di codice 1 viene riportato un esempio

di *Digital Twin Descriptor* riferito al contesto della *Smart Home* precedentemente definito, al fine di utilizzarlo come riferimento per illustrare i diversi possibili elementi del descrittore stesso.

```
{
  "id": "air-conditioner",
  "physical_asset": "air-conditioner01",
  "@context": "www.wodt.com/dtd",
  "@type": "device",
  "properties": {
    "current-temp": {
      "type": "double",
      "forms": [
        {
          "op": "readproperty",
          "href": "http://www.wodt.com/air-conditioner/current-temp"
        },
        {
          "op": "observeproperty",
          "href": "ws://www.wodt.com/air-conditioner/current-temp/observe"
        }]}
  },
  "events": {
    "functioning": {
      "data": "text/plain",
      "forms": [
        {
          "op": "subscribeevent",
          "href": "ws://www.wodt.com/air-conditioner/functioning/subscribe"
        }]}
  },
  "actions": {
    "set-switch": {
      "forms": [
        {
          "op": "invokeaction",
          "href": "http://www.wodt.com/air-conditioner/set-switch"
        }]}
  },
  "relationships": {
    "device_of": {
      "forms": [
        {
          "op": "readrelationship",
          "href": "http://www.wodt.com/air-conditioner/device_of"
        },
        {
          "op": "observerelationship",
          "href": "ws://www.wodt.com/air-conditioner/device_of/observe"
        }]}
  }
}
```

Listing 1: Esempio di un Digital Twin Descriptor per il DT di un condizionatore

Ogni *Digital Twin Descriptor* è articolato in 5 sezioni, che vengono analizzate singolarmente di seguito:

Informazioni di contesto : questa parte del documento, come parzialmente già detto, definisce un insieme di metadati che consentono sia di identificare il DT sia di contestualizzarlo nel suo dominio. Nello specifico, all'interno di questa sezione è possibile trovare i seguenti campi:

- `id`: adibito a riportare l'identificatore univoco associato al Digital Twin all'interno di WoDT.
- `@context`: chiave propria del formato JSON-LD, ha l'obiettivo di definire il dominio semantico in cui è collocato il Digital Twin. Di fatto, consente di specificare l'ontologia utilizzata per definire i nomi delle diverse componenti del descrittore stesso. In questo modo, le diverse applicazioni possono comprendere i termini utilizzati e interagire in maniera efficiente e accurata con il DT.
- `@type`: elemento chiave del formato JSON-LD, identifica il tipo semantico dell'entità fisica modellata dal Digital Twin. Solitamente, il tipo sarà scelto all'interno dell'ontologia specificata nel campo `@contex` e consentirà di classificare il DT stesso. Questo campo risulta particolarmente rilevante per poter realizzare *query* complesse e articolate.
- `physical_asset`: definisce l'identificatore dell'entità fisica virtualizzata dal Digital Twin. Nel caso in cui il DT modelli una composizione di altri DT, in questo campo sarà riportata la lista dei loro id. Questo campo è l'unico di quelli appartenenti a questa sezione che non è presente né nella TD né nel DTDL.

Proprietà (`properties`): questa chiave del documento identifica l'insieme delle proprietà del DT. Secondo il modello WoDT, tutte le proprietà possono esclusivamente essere lette e non modificate, cioè sono accessibili in modalità *read-only*, contrariamente alle proprietà definite dalla TD e dal DTDL. Un'altra caratteristica peculiare delle *properties* definite nel descrittore è data dal fatto che queste sono sempre osservabili, dunque, un'applicazione può sempre sottoscriversi ad una proprietà per ricevere una notifica ogniqualvolta

questa venga aggiornata. Questa operazione non è sempre disponibile per le proprietà definite da una TD, perchè dipende strettamente dalle capacità del *device* modellato. Nel caso del DTDL, invece, il concetto di osservazione delle *properties* non è direttamente inglobato in esse ma deve essere realizzato sfruttando il concetto di *telemetry*, creando confusione tra i due componenti. Similmente a quanto definito in una *Thing Description*, ciascuna proprietà del descrittore è identificata da una chiave ed è caratterizzata da un campo `type`, che permette di definire il tipo del valore modellato dalla proprietà stessa, e da una lista di `form` per la lettura e la sottoscrizione alla proprietà.

Eventi (`events`): questa sezione identifica l'insieme degli eventi di dominio che possono essere generati dal DT. Similmente alle proprietà, anche gli eventi possono essere osservati delle applicazioni che si sottoscrivono. La modalità con cui deve avvenire la sottoscrizione è definita dal `form` associato allo specifico evento, il quale è identificato da una chiave. Attraverso gli eventi, emerge una prima forma di *augmentation* da parte del DT, il quale, in quanto entità attiva, è in grado di applicare delle politiche di *monitoring* dei dati ricevuti dal PA, al fine di generare eventi di dominio. Il concetto di evento viene modellato anche nella TD mentre è assente nel DTDL, in favore del concetto, già citato, di *telemetry*.

Azioni (`actions`): attraverso questa chiave viene descritto l'insieme delle azioni invocabili tramite il Digital Twin. Come per le altre componenti, ogni azione è identificata da una chiave e ha associato un `form` che definisce come invocarla. Il concetto di azione è modellato negli stessi termini anche dalla *Thing Description* mentre è assente nel DTDL.

Relazioni (`relationships`): in corrispondenza di quest'ultima chiave è definito l'insieme delle relazioni che il Digital Twin è in grado di stabilire, cioè delle sue relazioni uscenti. Ciascuna relazione può essere o letta o osservata, nel primo caso si ottiene la lista delle istanze attive per tale tipo di relazione, nel secondo caso, invece, si ottene un aggiornamento quando il DT stabilisce una nuova istanza della relazione. Le relazioni sono identificate da un nome e definiscono il tipo semantico del proprio target e una lista di `form` che consentono la lettura e l'osservazione della relativa *relationships*. Il

concetto di relazione è assente nelle TD mentre è presente nei modelli definiti attraverso il DTDL ma, come per le proprietà, queste non possono essere osservate.

Un elemento chiave del descrittore è il concetto di `form`, ripreso dalla specifica di WoT. Un form definisce come realizzare uno specifico tipo di operazione in un determinato contesto, definendo il tipo di richiesta da eseguire e il target a cui rivolgerla. Il contesto del form consiste nell'elemento del descrittore in cui viene definito, può essere una proprietà, un evento, un'azione o una relazione, mentre, il tipo di operazione che può essere eseguita e il target a cui sottomettere la richiesta sono specificati attraverso due appositi campi:

- `op`: definisce in modo semantico il tipo di operazione descritta dal `form` stesso. Questo campo consente alle applicazioni di scegliere quale tra i molteplici `form` associati ad un componente, ad esempio ad una proprietà, è quello che permette di eseguire la funzionalità desiderata. Questo attributo può essere assegnato con uno più verbi che sintetizzano l'interazione semantica dell'operazione stessa. Ad esempio, nel form che definisce come leggere una determinata proprietà, questo campo presenterà il valore "`readproperty`";

- `href`: riporta l'URI della risorsa a cui deve essere rivolta la richiesta che permette di eseguire l'operazione di riferimento.

Il metodo HTTP da utilizzare per eseguire la richiesta, solitamente, non viene esplicitato ma a seconda dei casi si hanno dei valori impliciti di default. Ad esempio, per la lettura del valore di una proprietà si assume di default l'utilizzo del metodo GET mentre per l'invocazione di un'azione, che corrisponde all'operazione `invokeaction`, si considera sottinteso l'utilizzo del metodo POST che, infatti, consentirebbe anche l'invio dei dati necessari per eseguire l'azione stessa.

Il documento JSON in cui viene racchiuso il descrittore del Digital Twin sarà esposto dal WoDT Digital Adapter sulla rotta di default. L'*adapter* si occuperà poi di definire una rotta per ogni URI riportato nel documento stesso e per ciascuna implementerà l'operazione di riferimento.

4.3 Modelli di interazione

Al fine di chiarire le modalità con cui i diversi componenti dell'architettura di un *WoDT Platform Node* interagiscono tra loro e con quelli dei Digital Twin e quali sono i rapporti con le applicazioni esterne, in questa sezione vengono descritti alcuni diagrammi di sequenza relativi alle principali funzionalità della piattaforma individuate in fase di analisi.

4.3.1 Creazione di un Digital Twin

L'operazione di creazione di un Digital Twin è l'attività fondamentale per poter generare un ecosistema. Tale operazione può essere richiesta dal proprietario "fisico" del DT, ad esempio attraverso un applicativo, oppure può essere invocata da un Digital Twin già inserito nell'ecosistema. In figura 4.4 viene riportato lo schema delle interazioni che avvengono tra i diversi componenti della piattaforma e del DT, nel momento in cui il proprietario del DT esegue una richiesta di inserimento. L'operazione di creazione si sviluppa come segue:

1. **Richiesta di inserimento**: l'*owner* di un DT può eseguire una richiesta di inserimento, e dunque di creazione, di un Digital Twin, attraverso l'API esposta dalla *Platform External Interface*. Quest'ultima, una volta ricevuta la richiesta, provvede a direzionarla al *Digital Twins Manager* (DTM), il quale, dopo aver verificato la correttezza della richiesta, si occupa di gestirla. La richiesta di inserimento deve essere contenuta in un file JSON che contiene le seguenti informazioni: l'identificativo dell'*owner*, l'identificativo dell'immagine del Digital Twin, un id per il DT scelto dall'*owner* e una lista di tag per la classificazione del DT.

2. **Risposta alla richiesta di inserimento**: Ricevuta la richiesta, in primo luogo, il DTM si occupa di generare una risposta da inviare all'owner. Nello specifico, genera una risorsa che include: lo stato del processo di inserimento, l'identificativo della richiesta e l'indirizzo assegnato al DT. Inizialmente tale risorsa presenterà lo stato "`ACCEPTED`", se la richiesta è stata formulata correttamente, altrimenti la richiesta sarà rifiutata e la risorsa avrà stato "`REFUSED`". Per quanto riguarda

l'indirizzo del DT, questo sarà assegnato nel momento in cui il DT sarà effettivamente eseguito e inserito nell'ecosistema, momento in cui, inoltre, la risorsa passerà allo stato "`FULFILLED`". La risorsa generata dal DTM sarà inviata all'owner, il quale potrà rimanere in *polling* su di essa per determinare quando la sua richiesta è stata completata.

3. **Esecuzione del Digital Twin**: Una volta generata la risposta alla richiesta di inserimento, il processo procede in modo asincrono. Il DTM si occupa di eseguire il DT, avviando il relativo container, e poi procede a gestire le altre attività. Nel momento in cui il DT si avvia e attiva il proprio processo di *shadowing*, quest'ultimo richiede alla Management Interface (MI) di notificare al nodo della piattaforma che il DT è correttamente in esecuzione.

4. **Messaggio di Running**: La MI, sotto richiesta della funzione di *shadowing*, invia alla *Platform Internal Interface* il messaggio di *running*. Quest'ultimo notifica a tutti gli effetti al DTM che il DT è ufficialmente inserito nell'ecosistema e nel corpo del messaggio viene riportato il DT-Descriptor. Il DTM procede dunque, ad aggiornare la risorsa che identifica la richiesta di inserimento e notifica al *Registry Manager* e al *Graph Engine* che un nuovo DT è stato inserito nell'ecosistema. Il *Registry Manager* inserirà tutte le informazioni relative al DT nel suo elenco mentre il Graph Engine si occuperà di inserire il DT nella sua rappresentazione del grafo.

Nel caso in cui la richiesta di inserimento sia realizzata da un DT, le operazioni saranno le medesime, l'unica differenza riguarda l'interfaccia che accoglierà la richiesta di inserimento. In questo caso, infatti, la MI del DT padre, per ordine della funzione di *shadowing*, invierà la richiesta di inserimento, formattata come descritto precedentemente, alla *Platform Internal Interface*. L'interfaccia provvederà a consegnare la richiesta al DTM e, dunque, ad avviare il processo di creazione come appena descritto.

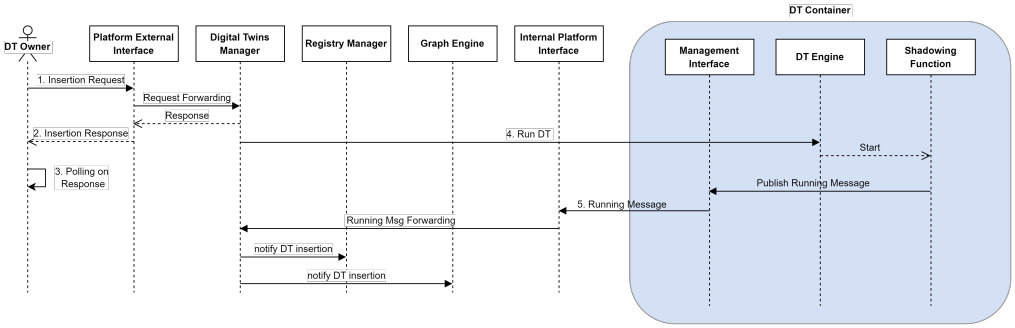


Figura 4.4: Creazione di un DT a fronte della richiesta del suo owner

4.3.2 Lettura e osservazione di una proprietà

In figura 4.5 sono riassunte le interazioni che consentono ad un'applicazione esterna di leggere il valore attuale di una proprietà e di osservarla. Nello specifico, viene mostrata anche l'operazione di *discovery* del DT che l'applicazione desidera contattare. Tale operazione inizia con la richiesta (1) da parte dell'applicazione alla *Platform External Interface* di tradurre l'id del Digital Twin nel relativo URI. L'interfaccia della piattaforma ridirezionerà tale richiesta al *Registry Manager*, il componente adibito a realizzare il servizio di ricerca. Ottenuto l'URI del DT, effettuando una richiesta a quest'ultimo (2), l'applicazione è in grado di ottenere il DT-Descriptor. Attraverso il descrittore, l'applicazione determina come poter leggere ed osservare la proprietà di interesse.

L'operazione di lettura di una proprietà, generalmente, prevede che un'applicazione esegua una richiesta GET sulla risorsa associata alla proprietà di interesse (3), come specificato dal descrittore. Il *WoDT Digital Adapter* provvederà a gestire tale richiesta consultando lo stato del Digital Twin al fine di determinare l'attuale valore della proprietà. Una volta recuperato il valore, quest'ultimo sarà restituito all'applicazione che ne ha fatto richiesta.

Un'applicazione può osservare una proprietà, eseguendo l'apposita richiesta indicata dal form di tale proprietà (4). Quando il *physical asset* notifica al *physical adapter* l'aggiornamento della proprietà osservata (5), tale evento e_{PA} sarà gestito dalla funzione di *shadowing*, la quale aggiornerà lo stato del Digital Twin secondo il suo modello. L'aggiornamento dello stato comporterà la generazione di un evento e_{DT} che il *WoDT Digital Adapter* invierà

(6) a tutte le applicazioni che stanno osservando la proprietà interessata dall'aggiornamento.

L'operazione di osservazione avviene con le stesse modalità anche per gli eventi e le relazioni. La navigazione di una relazione può essere equiparata alla lettura di una proprietà, in quanto, la richiesta avviene nello stesso modo. In tal caso, però, la risposta inviata conterrà la lista degli URI dei DT target di tale tipo di relazione.

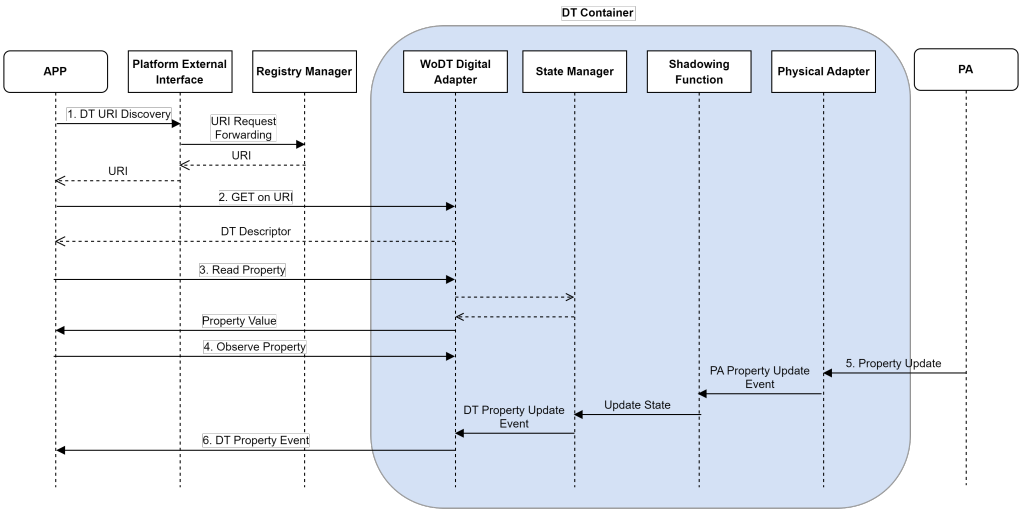


Figura 4.5: Lettura e osservazione di una proprietà da parte di un'applicazione

4.3.3 Invocazione di un'azione

L'invocazione di un'azione ad opera di un'applicazione esterna determina una serie di interazioni che coinvolgono unicamente le componenti interne del DT su cui è stata invocata. Anche per questa tipologia di interazione, la piattaforma viene al più, se necessario, contattata per il suo servizio di *discovery*. Nel diagramma in figura 4.6, vengono riportate dapprima le interazioni per ottenere il descrittore del DT a partire dal suo identificatore, spiegate precedentemente, e in seguito vengono rappresentate le diverse richieste che consentono l'operazione analizzata.

L'invocazione di un'azione ha origine da un'applicazione esterna che esegue la richiesta specificata dall'apposito form dell'azione di interesse (3). La

richiesta viene, anche in questo caso, gestita dal *WoDT Digital Adapter*, il quale si occuperà di generare l'evento e_{DT} associato (4). L'evento generato sarà elaborato, secondo il modello del DT, dalla funzione di *shadowing*. Generalmente, l'elaborazione comporterà la creazione di un evento e_{PA} che il *Physical Adapter* dovrà inviare al *physical asset* traducendolo nel tipo di messaggio adottato dal loro protocollo di comunicazione.

L'esecuzione dell'azione da parte del *physical asset* può comportare in quest'ultimo una modifica del suo stato (5), ad esempio, il valore di una proprietà potrebbe variare. In tal caso, si avranno le medesime interazioni descritte nel caso dell'aggiornamento di una proprietà osservata. Nel diagramma riportato, l'applicazione viene notificata dell'aggiornamento dello stato del Digital Twin, ma ciò avviene solo se l'aggiornamento coinvolge una proprietà, un evento o una relazione che è osservata dall'applicazione stessa.

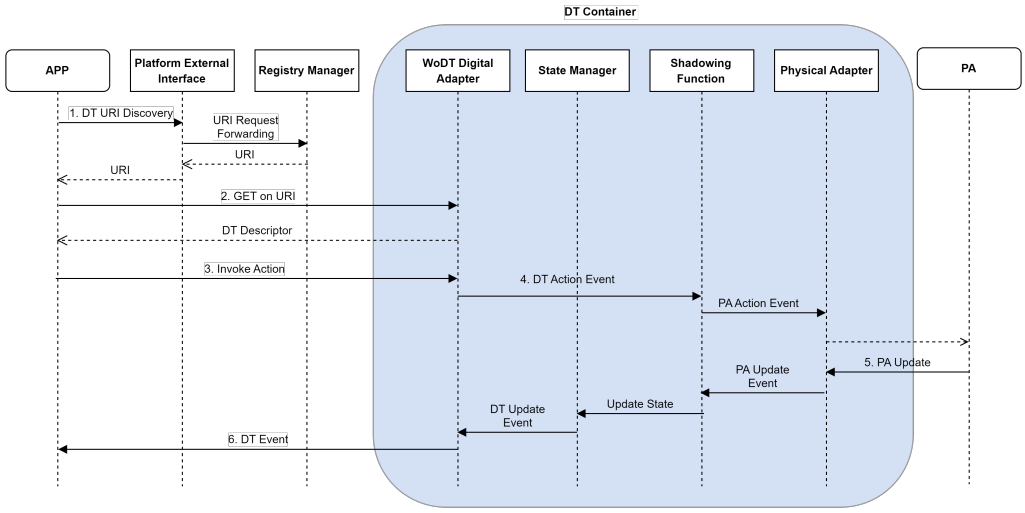


Figura 4.6: Invocazione di un'azione da parte di un'applicazione

4.3.4 Gestione dinamica delle relazioni

L'istanziazione di una relazione tra due Digital Twin presenti nell'ecosistema può avvenire, principalmente, per due motivazioni: il *physical asset* stabilisce una relazione concreta con un'altra entità del mondo fisico e il processo di *shadowing* riflette tale evento sullo stato del Digital Twin oppure è

la funzione di *shadowing* che seguendo il modello del Digital Twin determina l'esistenza di una relazione con un altro Digital Twin.

All'interno dello stato del DT, occorre differenziare il concetto di relazione da quello di istanza di una relazione. Nel primo caso, si fa riferimento ad un concetto semantico, ogni relazione attraverso il suo nome e il tipo semantico del suo target determina la diversa tipologia di legame che il DT è in grado stabilire. L'istanza di una relazione rappresenta, invece, il legame concreto presente tra il DT che la stabilisce e il DT target. Ad esempio, nel caso della *Smart Home* il DT *Bedroom* potrà avere nel suo modello due *relationship*, una con nome "`is_room_of`" e un'altra denominata "`has_device`". Un'istanza della prima tipologia di relazione potrà, ad esempio, avere come target il DT della casa, mentre, la relazione "`has_device`" potrà avere molteplici istanze, una per ciascun *device* presente nella camera. Un esempio di possibile istanza è quella che ha come target il DT *Air Conditioner*.

Nello stato di un DT ogni istanza di una relazione deve fare riferimento all'URI del DT che è oggetto dell'istanza stessa. Attraverso lo stato di un DT, invece, non sarà mai possibile determinare se tale DT è oggetto di un'istanza di una relazione di un altro DT.

In figura 4.7, sono evidenziate le interazioni che consentono la creazione di un'istanza di una relazione, a seguito della notifica da parte dell'entità fisica. Il processo ha inizio con la notifica al DT, da parte del *physical asset*, che quest'ultimo ha stabilito una relazione nel mondo fisico(1). Ricevuto l'evento e_{PA}, la funzione di *shadowing* lo elabora secondo il modello del DT e determina, eventualmente, quale tipologia di relazione del modello è stata istanziata. La funzione di *shadowing* può richiedere alla piattaforma, attraverso la MI, l'URI associato al DT oggetto della relazione(2). Ottenuto l'URI, l'istanza della relazione può essere creata e aggiunta allo stato del DT(3) e, successivamente, notificata alla piattaforma(4). Nello specifico, l'aggiornamento viene inviato dalla *Management Interface* alla *Platform Internal Interface*, la quale poi si occupa di notificare il *Graph Engine*(5). Quest'ultimo provvederà, dunque, ad aggiornare coerentemente la sua copia del grafo dell'ecosistema. Il messaggio che il DT invia alla piattaforma per notificare una nuova istanza di una relazione conterrà l'URI del DT che stabilisce l'istanza stessa, il nome della relazione di riferimento e l'URI del DT oggetto della relazione.

Per quanto riguarda l'eliminazione di un'istanza di una relazione, le mo-

tivazioni dell'evento sono le medesime definite per la creazione: è il *physical asset* che interrompe fisicamente il legame o è la funzione di *shadowing* che determina che tale legame non sussiste più. In ogni caso, la gestione dell'eliminazione avviene con lo stesso flusso di interazioni generato per la creazione. Ovviamente, lo stato del Digital Twin e del grafo gestito dal *Graph Engine* saranno aggiornati di conseguenza, cioè l'istanza interessata sarà rimossa.

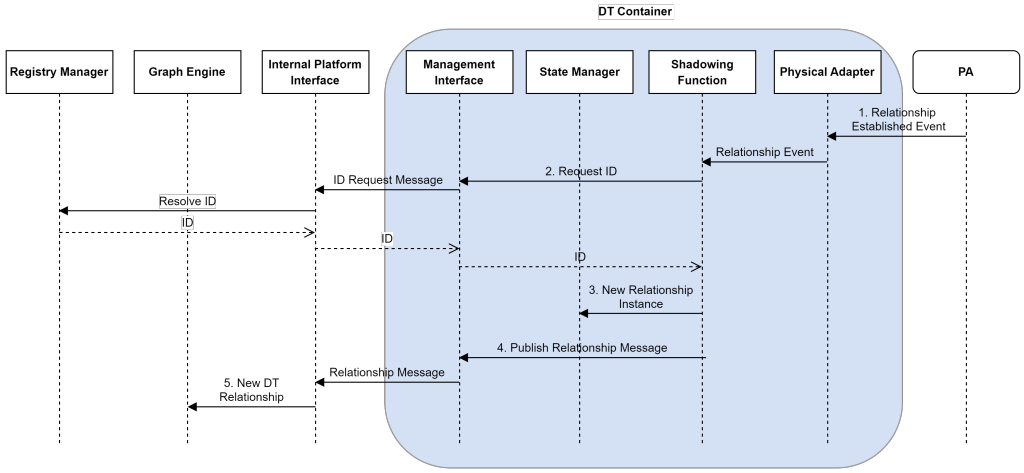


Figura 4.7: Creazione di un'istanza di una relazione

Capitolo 5

Implementazione

In questo quinto capitolo è descritta la fase di implementazione di questo progetto di tesi. Nello specifico, l'implementazione è stata condotta attraverso 3 fasi successive: in primo luogo si è estesa la libreria White Label Digital Twin (WLDT) aggiungendo le componenti essenziali per sviluppare DT attinenti al modello WoDT, dopodiché si è realizzato il primo prototipo del *WoDT Platform Node*, in grado di gestire ecosistemi di DT definiti attraverso la libreria citata, e infine si è sviluppato il *WoDT Platform Visualizer*, un client web che consente di visualizzare lo stato generale dell'ecosistema.

Il capitolo presenta le 3 fasi sopracitate nello stesso ordine con cui sono state realizzate, dunque viene prima descritta l'implementazione dei Digital Twin, successivamente quella del *WoDT Platform Node* e infine quella del *WoDT Platform Visualizer*. Per ogni sezione vengono definite le tecnologie utilizzate e gli aspetti fondamentali della loro realizzazione.

Al termine del capitolo viene presentato un esempio di utilizzo del prototipo realizzato.

5.1 Implementazione dei Digital Twin

L'obiettivo della fase di implementazione dei Digital Twin è stato quello di estendere il framework Java White Label Digital Twin (WLDT) al fine di produrre una versione della libreria che consentisse la realizzazione di DT fedeli al modello proposto in WoDT e in grado di integrarsi con la piattaforma da sviluppare.

WLDT, nella sua versione 2.0, adotta già parzialmente l'architettura di DT proposta in WoDT, cioè, presenta i concetti di Physical Adapter e Digital Adapter, modella l'idea di funzione di *shadowing* e di stato del Digital Twin, e si basa su un'architettura ad eventi guidata dal WLDT Engine.

Nella sua versione base, però, sono assenti moduli fondamentali come il WoDT Digital Adapter e la Management Interface e nella modellazione dello stato del DT non è presente il concetto di relazione, essenziale per poter realizzare un *Web of Digital Twins*. Nel contesto di questa tesi si è dunque estesa la libreria per integrare le componenti mancanti.

5.1.1 Panoramica White Label Digital Twin

La libreria di riferimento è stata realizzata con lo scopo di produrre un framework Java che consentisse di semplificare il più possibile la progettazione e lo sviluppo dei Digital Twin, al fine di integrarli ad applicazioni nell'ambito dell'Internet of Things.

Il framework è stato progettato con l'obiettivo di massimizzare la modularità, la flessibilità e il riuso delle varie componenti al fine di definire in modo efficace le controparti digitali di oggetti fisici. La caratteristica peculiare della libreria è quella di essere sufficientemente generica e adattabile da supportare la definizione di una replica digitale per ogni tipologia di entità fisica, in modo da estendere le sue funzionalità, ad esempio, supportando ulteriori protocolli di comunicazione o traducendo e normalizzando i dati prodotti. I tre requisiti che hanno guidato la realizzazione di WLDT sono, per l'appunto, semplicità, estensibilità e portabilità, il quale implica una visione a microservizi.

Un'istanza di WLDT può essere considerata un agente software che implementa tutte le caratteristiche e le funzionalità di un Digital Twin, il quale può essere eseguito in Cloud come in Edge. I Digital Twin generati attraverso WLDT, infatti, sono ciascuno un microservizio facilmente eseguibile attraverso un container Docker.

In figura 5.1 sono rappresentate le componenti principali che costituiscono l'architettura di WLDT e, dunque, attraverso le quali si implementa il singolo Digital Twin. Nello specifico, dall'immagine è possibile individuare i tre livelli su cui si sviluppa l'architettura: quello relativo al *core* della libreria, quello che modella il DT e, infine, quello degli *adapter*.

Di seguito, ciascun modulo viene descritto brevemente analizzando le sue caratteristiche fondamentali.

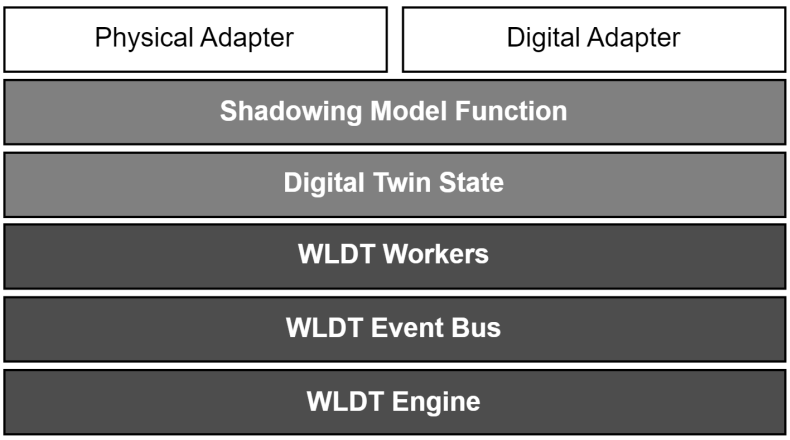


Figura 5.1: Rappresentazione delle componenti dell'architettura di White Label Digital Twin

WLDT Engine Costituisce il motore *multi-thread* della libreria. Consente, infatti, di eseguire e monitorare molteplici *workers* contemporaneamente. Il WLDT Engine si occupa, dunque, di orchestrare i diversi moduli interni dell'architettura mantenendo i riferimenti di ognuno. Può essere considerato il motore del DT stesso.

WLDT Event Bus Rappresenta l'Event Bus interno, progettato per supportare la comunicazione tra le diverse componenti dell'istanza. Consente di definire eventi personalizzati per modellare sia quelli fisici (e_{PA}) che quelli digitali (e_{DT}). Ogni modulo può definire un Event Filter al fine di specificare quali tipologie di eventi è interessato a gestire, associando a ciascuna un'apposita *callback* per elaborare i diversi messaggi.

WLDT Workers Modella il componente interno di base; costituisce, infatti, il singolo elemento eseguibile dal WLDT Engine. Escluso il Digital Twin State, ciascuno dei moduli descritti successivamente definisce un'implementazione specifica di un WLDT Workers.

Digital Twin State Struttura lo stato del DT definendo la lista delle proprietà, degli eventi e delle azioni. Le diverse istanze inserite nelle liste possono corrispondere direttamente a degli elementi del *physical asset* o possono derivare da una loro combinazione, in ogni caso è la Shadowing Model Function (SMF) che definisce il *mapping*, seguendo il modello definito dal progettista. Questo componente espone, inoltre, un insieme di metodi per consentire alla SMF la sua manipolazione. Ogni volta che il Digital Twin State viene modificato, quest'ultimo si occupa di generare il corrispondente e_{DT}.

Shadowing Model Function Responsabile di definire il comportamento del Digital Twin interagendo con il Digital Twin State. Precisamente, implementa il processo di *shadowing* che consente di mantenere sincronizzato il DT rispetto alla sua entità fisica. Questo componente si basa su una specifica implementazione di un WLDT Workers chiamata Model Engine, al fine di essere eseguito dal WLDT Engine. La Shadowing Model Function costituisce il componente fondamentale che deve essere esteso dal progettista del DT per concretizzarne il modello. La funzione di *shadowing* osserva il ciclo di vita del Digital Twin al fine di essere notificata dei diversi cambiamenti di stato. Ad esempio, viene informata quando il DT passa allo stato *Bound*, cioè quando i suoi Physical Adapter hanno terminato la procedura di *binding* con l'*asset* fisico. Questo componente, inoltre, consente al progettista di definire il comportamento del DT nel caso in cui venga modificata una proprietà, scatenato un evento o invocata un'azione.

Physical Adapter Definisce le funzionalità essenziali che le singole estensioni, relative agli specifici protocolli, devono implementare. Come previsto dal modello di WoDT, un DT può essere dotato di molteplici Physical Adapter al fine di gestire la comunicazione con la relativa entità fisica. Ognuno produrrà una Physical Asset Description (PAD), cioè una descrizione delle proprietà, degli eventi, delle azioni e delle relazioni che l'*asset* fisico espone attraverso lo specifico protocollo. Il DT esegue il passaggio di stato da *Unbound* a *Bound* quando tutti i suoi Physical Adapter hanno prodotto il relativo PAD. La Shadowing Model Function, seguendo il modello del DT, selezionerà le componenti dei diversi PAD che è interessata a gestire.

Digital Adapter Fornisce l'insieme delle callback che ciascuna implementazione specifica può utilizzare per essere notificata dei cambiamenti di stato del DT. Simmetricamente a quanto avviene per i Physical Adapter, un Digital Twin può definire molteplici Digital Adapter per esporre il proprio stato e funzionalità attraverso diversi protocolli.

Dunque, per realizzare un Digital Twin attraverso WLDT è necessario definire una Shadowing Model Function e almeno un Physical Adapter e un Digital Adapter, in modo tale da consentire la connessione con l'entità fisica e abilitare il DT ad essere utilizzato dalle applicazioni esterne. Definite le 3 componenti, è possibile istanziare il WLDT Engine e, successivamente, avviare il ciclo di vita del DT.

5.1.2 Estensioni realizzate

La prima parte della fase di implementazione del presente progetto di tesi ha riguardato l'estensione della libreria WLDT. Al fine di realizzare dei test di integrazione più completi, sono stati realizzati i due *adapter* MQTT e un primo prototipo del Digital Adapter basato su HTTP.

Successivamente, sono state sviluppate le componenti necessarie per realizzare una versione della libreria totalmente adattata al modello WoDT e interoperabile con la piattaforma da implementare. Nello specifico, sono state aggiunte le relazioni allo stato del Digital Twin e definite la Management Interface, la WoDT Shadowing Function e il WoDT Digital Adapter.

MQTT Physical Adapter e MQTT Digital Adapter

Entrambi gli adapter basati sul protocollo MQTT sono stati realizzati attraverso l'utilizzo della libreria Eclipse Paho, la quale consente di definire facilmente un MQTT Client.

Il protocollo MQTT è uno standard per lo scambio di messaggio in applicazioni IoT, infatti, è stato progettato per essere estremamente efficiente e leggero. Il protocollo si basa sul meccanismo *Publish/Subcribe* e la comunicazione tra i vari MQTT Client è mediata da un MQTT Broker.

Un client, per ricevere i messaggi, deve sottoscriversi ad almeno un topic presso il broker. Quando un client pubblica un messaggio su un determinato

topic, il broker si occupa di inviare una copia del messaggio a tutti i client sottoscritti a tale topic.

L'implementazione dei due adapter è stata realizzata in modo speculare utilizzando due concetti base:

- **Outgoing Topic**: rappresenta un topic MQTT "uscente" rispetto al DT stesso, cioè è un topic sul quale il DT, o più nello specifico i due *adapter*, pubblicano i messaggi al fine di comunicare con il client Target.

- **Incoming Topic**: rappresenta un topic MQTT "entrante" rispetto al DT stesso, cioè è un topic rispetto al quale il DT, o per meglio dire i suoi *adapter*, si sottoscrivono, al fine di ricevere i messaggi dai client.

Per quanto riguarda l' MQTT Physical Adapter, il client di riferimento è l'*asset* fisico, dunque, sono topic entranti tutti quelli sui quali l'entità fisica pubblica messaggi relativi al suo stato, mentre sono degli Outgoing Topic quelli utilizzati dal *physical asset* per ricevere i comandi.

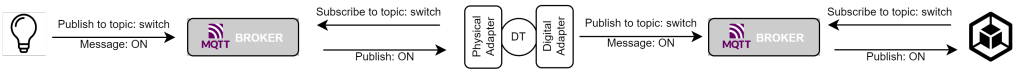
In modo speculare, per l' MQTT Digital Adapter, il quale ha come client le diverse applicazioni che interagiscono con il DT, gli Icoming Topic sono quelli che gestiscono i messaggi di invocazione delle azioni mentre gli Outgoing Topic sono quelli attraverso i quali vengono pubblicati i messaggi relativi all'aggiornamento delle proprietà e degli eventi.

Tra gli Incoming Topic del Physical Adapter e gli Outgoing Topic del Digital Adapter non esiste necessariamente un mapping uno a uno e non vi è nessun vincolo circa i loro nomi. L'unica relazione tra le due tipologie di topic è data dal processo di *shadowing* che definisce come le proprietà dello stato dell'entità fisica siano tradotte in quelle del Digital Twin.

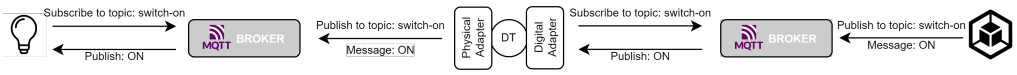
Entrambe le tipologie di topic consentono al progettista di definire il nome del topic stesso e il tipo del corpo del messaggio. Gli Outgoing Topic offrono, inoltre, un metodo per serializzare il messaggio da inviare, mentre, gli Incoming Topic consentono l'operazione di deserializzazione.

Entrambi gli *adapter*, al fine di essere totalmente generici e non dipendere da modelli di DT predefiniti, sono stati progettati per essere estremamente configurabili. Il progettista di un DT per utilizzare uno dei due *adapter* dovrà definire: l'MQTT Broker da utilizzare, l'insieme dei topic entranti, specificando per ognuno il nome e la relativa funzione di traduzione del messaggio,

e l'insieme dei topic uscenti, indicando il nome e la funzione di serializzazione per ciascuno.



(a) Aggiornamento di una proprietà attraverso il protocollo MQTT



(b) Invocazione di un'azione attraverso il protocollo MQTT

Figura 5.2: Utilizzo degli *adapter* MQTT

In figura 5.2a, sono riassunte le interazioni che avvengono per gestire l'aggiornamento del valore di una proprietà. Come si può notare, il Physical Adapter si sottoscrive al topic della proprietà, il quale viene utilizzato dall'entità fisica per pubblicare gli aggiornamenti. Dall'altro lato, il Digital Adapter pubblica su un topic con il medesimo nome, un messaggio relativo al cambiamento di stato del DT, il quale sarà ricevuto da tutte quelle applicazioni precedentemente sottoscritte al topic.

In figura 5.2b, è riportato un esempio speculare relativo all'invocazione di un'azione da parte di un'applicazione esterna. In questo caso, il Digital Adapter riceve sul suo Incoming Topic `switch-on` il messaggio di invocazione dell'azione. Tale messaggio sarà tradotto in uno specifico e_{DT} che la funzione di shadowing mapperà nel relativo e_{PA}, quest'ultimo sarà poi associato ad un messaggio dell'Outgoing Topic `switch-on` del Physical Adapter per inviarlo all'entità fisica. Quest'ultima, essendosi precedentemente sottoscritta al medesimo topic, sarà in grado di ricevere ed elaborare il comando di accensione.

HTTP Digital Adapter

Il Digital Adapter basato sul protocollo HTTP è stato progetto con l'obiettivo di realizzare un primo prototipo del futuro WoDT Digital Adapter. Questo adapter è stato implementato per poter consentire alle applicazioni esterne di fruire del DT come se fosse un servizio REST ma senza introdurre il concetto fondamentale di *Digital Twin Descriptor*.

L'HTTP Digital Adapter è stato implementato utilizzando la libreria Undertow che consente di definire web server flessibili e performanti in Java. Grazie a questo primo prototipo, il Digital Twin offre delle rotte standard per ottenere la lista delle proprietà, degli eventi e delle azioni e supporta l'invocazione delle azioni esposte attraverso apposite richieste POST.

In figura 5.3, è riportata l'API esposta da un DT attraverso l'HTTP Digital Adapter. Ad esempio, un'applicazione che desidera conoscere le proprietà modellate dal DT *Air Conditioner* potrà effettuare la richiesta GET all'indirizzo `http://<digitalTwinAddress>/properies` e una volta determinata la chiave della proprietà che modella la temperatura attuale, potrà monitorarne direttamente il valore eseguendo richieste GET sulla rotta `http://<digitalTwinAddress>/properies/current-temp/value`. La medesima applicazione potrà, invece, accendere o spegnere il condizionatore attraverso richieste POST sulle apposite rotte. Ad esempio, se l'azione di accensione ha chiave `switch-on`, allora attraverso una POST all'indirizzo `http://<digitalTwinAddress>/actions/switch-on` l'applicazione sarà in grado di accendere il condizionatore fisico e potrà visualizzare l'effetto della sua richiesta richiedendo il valore della proprietà che modella lo stato di accensione o spegnimento.

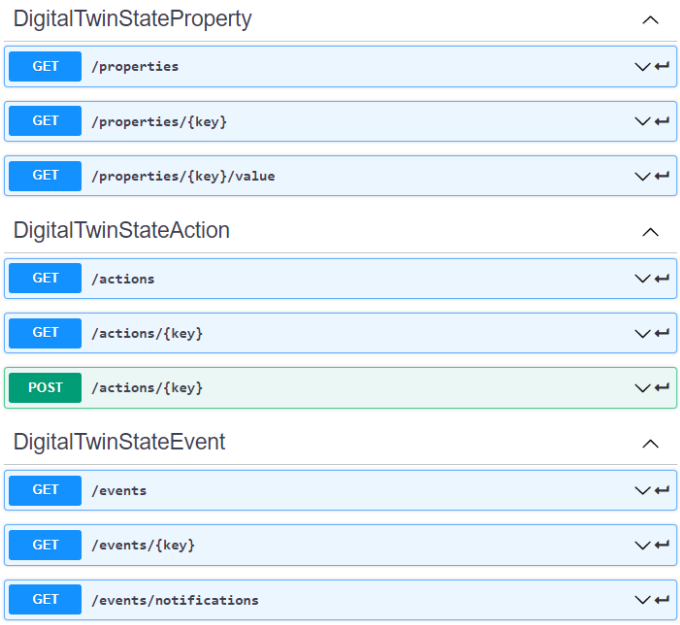


Figura 5.3: Documentazione dell'API esposta dall'HTTP Digital Adapter

Un altro aspetto fondamentale non incluso in questa prima versione del Digital Adapter basato su HTTP è l'osservazione delle componenti dello stato del Digital Twin, funzionalità inclusa, invece, nel WoDT Digital Adapter descritto successivamente.

Integrazione delle relazioni

Le relazioni sono uno dei concetti fondamentali per poter definire ecosistemi di Digital Twin e per poter modellare adeguatamente i rapporti presenti nella realtà tra le varie entità fisiche. Il concetto di relazione non era, però, incluso nella versione iniziale della libreria WLDT, dunque, in questo progetto di tesi, si è deciso di estenderla al fine di introdurlo.

Come descritto nel paragrafo 4.3.4, è necessario distinguere tra due concetti, quello di *Relationship* e quello di *Relationship Instance*: il primo modella la relazione dal punto di vista semantico, ne definisce il nome e la tipologia di target, mentre il secondo rappresenta un'istanziazione del concetto nella realtà. Ad esempio, nel contesto della *Smart Home*, il DT *Home* definirà la *Relationship* chiamata `has_room` che ha come possibili target i DT che rappresentano le diverse stanze della casa, mentre il legame effettivo tra il DT *Home* e quello *Bedroom* sarà modellato da un'apposita *Relationship Instance* della relazione `has_room`.

L'integrazione delle relazioni nel framework WLDT ha, quindi, richiesto l'implementazione dei due concetti di relazione e istanza e ha determinato la necessità di aggiornare lo stato del Digital Twin per includere la lista delle relazioni e allo stesso modo è stato modificata la Physical Asset Description, in modo tale da consentire ai Physical Adapter di specificare anche quali relazioni sono in grado di gestire.

Dato il requisito di poter gestire in modo dinamico le relazioni e considerato il modello ad eventi del framework, sono stati modellati due tipologie di eventi:

- **Relationship Instace Created**: modella l'evento di creazione di un'istanza di una relazione nel mondo fisico, che, dunque, deve essere gestito dalla funzione di *shadowing* per essere riflesso nello stato del Digital Twin in modo concorde al suo modello.

- **Relationship Instace Deleted**: modella la possibilità che una relazione tra due entità termini e che quindi la relativa istanza debba essere eliminata. Anche in questo caso è la funzione di *shadowing* che determina come gestire questo evento rispetto al modello del Digital Twin.

Al fine di gestire adeguatamente tali eventi è stata estesa l'implementazione del core degli adapter. In particolare, alla classe *Physical Adapter* sono stati aggiunti i metodi per notificare i due diversi eventi mentre al modello dei Digital Adapter sono state integrate le relative callback.

Similmente a quanto avviene nel caso dell'aggiornamento di una proprietà, l'entità fisica, quando stabilisce una nuova istanza di una relazione con un determinato *asset*, notifica dell'evento il Physical Adapter. Quest'ultimo, a sua volta, pubblicherà un evento di tipo Relationship Instance Created che deve essere gestito dal processo di *shadowing*. La Shadowing Model Function elaborerà l'evento ricevuto secondo il modello del DT definito dal progettista, questo potrà determinare o l'inserimento dell'istanza nello stato del DT o una modifica dello stato di qualche altro tipo o lo scarto dell'evento stesso. Se tale istanza viene aggiunta ad una relazione dello stato del DT, allora, il Digital Twin State stesso pubblicherà il relativo e_{DT} per notificare dell'evento tutti gli altri componenti. I Digital Adapter, attraverso l'apposita callback, saranno in grado di elaborare a loro volta l'evento e di renderlo disponibile, sotto una qualche forma, alle applicazioni.

L'attuale implementazione del concetto di relazione, in futuro, potrà essere ulteriormente estesa, ad esempio, includendo la possibilità di gestire delle proprietà associate alle relazioni o rafforzando l'aspetto semantico.

WoDT Digital Adapter

Il WoDT Digital Adapter può essere considerato un'evoluzione dell'HTTP Digital Adapter che include due aspetti fondamentali in ottica *Web of Digital Twins*: il Digital Twin Descriptor e la possibilità di osservare proprietà, eventi e relazioni.

Nel contesto WoDT, è richiesto che sia ciascun Digital Twin ad esporre il proprio descrittore in modo indipendente, al fine di consentire interoperabilità tra applicazioni appartenenti ad organizzazioni differenti. Inoltre,

ciascun DT deve essere fruibile in modalità *as-a-service*, dunque, è responsabilità del relativo WoDT Digital Adapter implementare il server web e esporre il descrittore.

Data la natura dei Digital Twin e la loro capacità di evolvere per adattarsi ai cambiamenti dell'entità fisica, anche il *Digital Twin Descriptor* deve essere prodotto in modo da poter riflettere le eventuali modifiche nel modello del Digital Twin. Per questo motivo, è responsabilità del WoDT Digital Adapter esporre sempre una versione aggiornata del descrittore sulla rotta di default. Nello specifico, il descrittore viene prodotto nel momento in cui viene richiesto e in base allo stato del Digital Twin.

Seguendo il modello di descrittore riportato nel paragrafo 4.2.1, per ciascuna proprietà, evento, relazione e azione presente nello stato del DT, il WoDT Digital Adapter genera l'apposita sezione del documento e i relativi form. Inoltre, è, ovviamente, responsabilità dell'adapter gestire le richieste formulate su ciascuna delle rotte definite dal descrittore stesso.

Per quanto riguarda l'osservazione delle componenti dello stato del DT, si è deciso di utilizzare il protocollo WebSocket che consente una comunicazione bidirezionale, basata su TCP, tra client e server. In questo modo, quando un'applicazione esegue la richiesta associata all'operazione di osservazione di una proprietà, relazione o evento, si aprirà una WebSocket tra il WoDT Digital Adapter e l'applicazione stessa. Dunque, quando tale proprietà sarà modificata, l'adapter potrà notificare di conseguenza l'applicazione. Nello specifico, si è deciso di utilizzare un modulo aggiuntivo della libreria Undertow che consente di integrare le WebSocket in applicativi Java.

Il WoDT Digital Adapter mantiene per ciascuna proprietà, relazione ed evento, una lista delle WebSocket aperte, così da identificare l'insieme delle applicazioni interessate ad osservare il relativo componente. In questo modo, quando il WoDT Digital Adapter riceve un evento che notifica l'aggiornamento di un componente, l'adapter stesso potrà inviare un messaggio su tutte le WebSocket registrate nella lista di tale componente e conseguentemente, notificare tutte le applicazioni interessate. L'applicazione che decide di terminare l'osservazione di un elemento dello stato del DT, potrà notificare il WoDT Digital Adapter di tale decisione e la relativa WebSocket sarà rimossa dalla lista associata all'elemento.

Questo componente fa parte del modulo aggiuntivo della libreria WLDT,

sviluppato in questo progetto di tesi, che consente di estendere la libreria stessa per poter definire dei Digital Twin fedeli al modello WoDT e integrabili alla piattaforma sviluppata e descritta successivamente.

Management Interface

Come descritto precedentemente, il ruolo della Management Interface, nell'architettura di un DT, è quello di consentire l'interazione con la *WoDT Platform*. Per questo motivo, già dalla fase di progettazione si era individuata la necessità di sviluppare questo componente come un client HTTP. Nello specifico, la libreria Java che si è deciso di utilizzare per implementare il componente è OkHttp, la quale semplifica la formulazione delle diverse richieste HTTP.

La decisione di utilizzare WLDT come framework di riferimento per l'implementazione dei DT ha determinato la necessità di sviluppare questo componente in modo tale da potersi adeguare al modello definito dalla libreria stessa. La realizzazione della Management Interface, inoltre, è stata guidata dalla volontà di rendere la sua integrazione quanto più possibile trasparente allo sviluppatore che utilizza la libreria. Ad esempio, uno dei compiti della Management Interface è quello di notificare alla piattaforma che il DT si è avviato correttamente, tale operazione non deve essere eseguita esplicitamente dal progettista che definisce la Shadowing Model Function ma grazie allo sviluppo combinato dalla MI e della WoDT Shadowing Function, descritta successivamente, avviene in automatico in modo trasparente.

La Management Interface è stata, dunque, sviluppata come un client HTTP che interagisce con l'Internal Platform Interface su richiesta dalla funzione di *shadowing*, unico componente all'interno dell'architettura del DT che ha motivo di relazionarsi con la *WoDT Platform*.

La MI espone i metodi per pubblicare il messaggio di *running* quando il Digital Twin passa allo stato *Bound*, per richiedere la traduzione di un identificativo di un DT nel corrispondente URI, per notificare la creazione e l'eliminazione di un'istanza di una relazione e per notificare la terminazione del ciclo di vita del DT.

WoDT Shadowing Function

La WoDT Shadowing Function è stata realizzata come un'estensione della Shadowing Model Function, presente nella versione base del framework WLDT, al fine di integrare la Management Interface sviluppata. Questo componente, infatti, richiede al progettista del DT di implementare le medesime callback presenti nella Shadowing Model Function ma incapsula anche i meccanismi di notifica alla *WoDT Platform*. Ad esempio, quando la funzione riceve l'evento di Relationship Instance Created, si attiva la relativa callback che consente al progettista di definire come integrare tale evento allo stato del Digital Twin, nella nuova versione della callback, se il progettista decide di inserire l'istanza nello stato del Digital Twin, allora dovrà restituirla come risultato della callback cosicché, in automatico, la Management Interface possa notificarla alla piattaforma.

La scelta di sviluppare questo componente consente di integrare l'utilizzo della Management Interface al framework WLDT, senza rendere quest'ultimo vincolato al modello WoDT. Infatti, la libreria nasce per essere *general purpose* e non per implementare unicamente Digital Twin in ottica WoDT.

Riassumendo, per poter definire, attraverso la libreria WLDT, un Digital Twin che adotti il modello *Web of Digital Twins* e si integri con la WoDT Platform, è necessario utilizzare il WoDT Digital Adapter, definire la WoDT Shadowing Function e sviluppare l'apposito Physical Adapter per relazionarsi con l'entità fisica. In figura 5.4, è riportata l'architettura del framework WLDT integrato al modulo per WoDT, sviluppato in questo progetto di tesi.

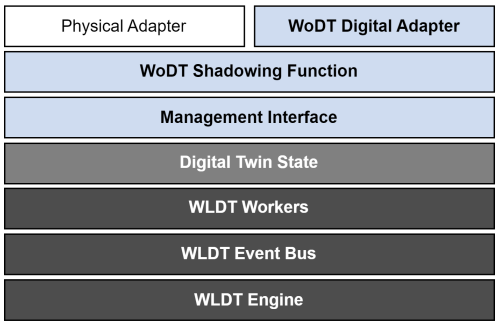


Figura 5.4: Rappresentazione dell'architettura della versione della libreria WLDT adattata a Web of Digital Twins

5.2 Implementazione della WoDT Platform

L'obiettivo principale della fase di implementazione del presente progetto di tesi è stato la realizzazione della WoDT Platform, cioè della piattaforma che supporta la realizzazione di ecosistemi di Digital Twin secondo il modello *Web of Digital Twins*.

Oltre ad un primo sviluppo prototipale del WoDT Platform Node, si è deciso di implementare anche un client Web che consenta di visualizzare graficamente lo stato dell'intero ecosistema e quello dei singoli DT, tale visualizzatore è stato per l'appunto denominato WoDT Platform Visualizer.

Di seguito, verranno presentate le tecnologie utilizzate e i dettagli implementativi sia per il WoDT Platform Node che del WoDT Platform Visualizer.

5.2.1 Implementazione del WoDT Platform Node

Per quanto riguarda l'aspetto tecnologico del WoDT Platform Node, il linguaggio di riferimento scelto è *Java* e la libreria utilizzata per l'implementazione dei server web, che costituiscono le interfacce di comunicazione del nodo, è *Undertow*, data la sua flessibilità e semplicità.

L'architettura finale del WoDT Platform Node rispetta fedelmente quella delineata in fase di progettazione, per quanto riguarda le componenti fondamentali per la creazione e gestione dell'ecosistema. Nel prototipo sviluppato è presente, inoltre, una terza interfaccia che consente la comunicazione tra il Visualizer e il nodo.

Il principio generale che ha guidato la fasi di implementazione della piattaforma è stato quello di separare, il più possibile, la logica applicativa dai protocolli di comunicazione scelti, in modo tale da renderla indipendente e facilitare una futura evoluzione della piattaforma.

Di seguito, sono descritti i principali dettagli implementativi per ciascuna delle componenti finali del WoDT Platform Node.

Platform External Interface

Nella fase di progettazione è stato deciso di implementare l'interfaccia di comunicazione esterna come un server web, a cui affidare la gestione delle richieste provenienti dal livello delle applicazioni. Nonostante questo, l'implementazione del componente è stata realizzata scomponendo gli aspetti

tecnici legati all'implementazione del server HTTP da quelli di smistamento delle richieste tra i componenti interni dell'architettura. In questo modo, tale interfaccia potrà essere facilmente implementata anche utilizzando protocolli di comunicazione differenti.

La logica applicativa, che consente di direzionare ciascuna richiesta al modulo interno responsabile di elaborarla, è stata incapsulata all'interno di una classe astratta, che deve essere estesa al fine di legarla ad uno specifico protocollo. La classe definisce un metodo per ciascuna delle richieste che la Platform External Interface supporta, come ad esempio, la richiesta di creazione dei DT e quelle relative alla funzionalità di *discovery*. Ciascun metodo si occupa, quindi, di notificare il modulo interno coinvolto dalla richiesta, ad esempio, nel caso dell'inserimento di un DT, i dati inviati dall'*owner* saranno forniti al Digital Twin Manager (DTM) al fine di attivare il processo di creazione e inserimento.

Nel prototipo realizzato, la classe astratta definita è stata concretizzata da un'implementazione legata al protocollo HTTP, che ha consentito di definire tale interfaccia come un server web REST.

Platform Internal Interface

Anche questa interfaccia è stata progettata e, dunque, implementata come un server web, che ha l'obiettivo di gestire le interazioni con i Digital Twin associati al WoDT Platform Node.

Questo componente, nonostante si relazioni con un livello differente, ha il medesimo ruolo della Platform External Interface, per questo motivo, è stata implementata seguendo le stesse scelte. In questo caso, inoltre, la possibilità di estendere i protocolli utilizzabili per comunicare con i DT è ancora più importante, considerando la loro eterogeneità in ottica di ecosistema aperto.

Similmente alla Platform External Interface, la classe astratta, a cui è affidata la logica di smistamento delle richieste ai moduli interni, mantiene i riferimenti dei 3 componenti interni responsabili di offrire le diverse funzionalità della piattaforma.

Digital Twins Manager

Il Digital Twins Manager è il componente centrale dell'architettura, in quanto è incaricato di gestire l'insieme dei Digital Twin per tutto il loro ciclo di vita.

L'implementazione di questo componente, in questo progetto di tesi, ha riguardato principalmente gli aspetti di gestione delle richieste dal punto di vista logico. L'aspetto di esecuzione automatica dei singoli Digital Twin, attraverso container Docker, ad esempio, è stata progettata ma non inclusa nella versione finale di questo componente.

Il Digital Twin Manager mantiene i riferimenti dei moduli interni che hanno la necessità di essere notificati quando un DT è stato inserito nell'ecosistema, in questa versione dell'architettura, tali moduli sono il Registry Manager e il Graph Engine. Al fine di consentire una futura evoluzione della piattaforma, il DTM mantiene una lista per registrare i *listener* dell'evento di inserimento di un Digital Twin, in questo modo, futuri componenti aggiuntivi potranno semplicemente essere inseriti in tale lista per essere notificati.

Il compito principale del DTM è quello di gestire le richieste di creazione dei DT e di eseguirli. Per questo motivo, il DTM mantiene uno storico di tutte le richieste di inserimento e tiene traccia dello stato di tutti i DT avviati in un apposito elenco.

In particolare, alla ricezione di una richiesta di inserimento, il DTM crea la relativa risorsa (Digital Twin Insertion Request) nel suo storico. Tale risorsa avrà stato `REFUSED` se la richiesta di inserimento non è stata formulata adeguatamente altrimenti sarà memorizzata con stato `ACCEPTED`. Se la richiesta viene accettata, il DTM assegna al DT un URI e aggiunge il relativo Digital Twin Container al suo elenco. Al client che ha realizzato la richiesta di creazione del DT, sarà restituita la Digital Twin Insertion Request in modo tale da poter saper se e quando l'inserimento del DT è andato a buon fine. Infatti, tale risorsa sarà aggiornata dal DTM quando quest'ultimo riceverà dal DT il messaggio di Running, nello specifico, la DT Insertion Request passerà allo stato '`FULFILLED`' e gli sarà inserito l'URI a cui contattare il DT creato. In questo modo, il processo di creazione è totalmente asincrono e l'owner del DT potrà monitorare lo stato della sua richiesta attraverso un polling sulla risorsa.

Alla ricezione del messaggio di *running*, il DTM notifica dell'evento tutti

i *listener* registrati nell'apposita lista. In particolare, il messaggio di notifica contiene quella che è stata definita Internal Digital Twin Description, la quale contiene: l'URI e l'identificativo del DT, i riferimenti all'*owner* e il Digital Twin Descriptor, contenuto nel messaggio di running.

Quando un DT termina il proprio ciclo di vita, notifica il DTM attraverso un messaggio di *Stop* che comporterà l'aggiornamento dello stato del Digital Twin Container memorizzato dal DTM e la contestuale notifica ai *listener*.

Registry Manager

Il compito del Registry Manager è quello di mantenere un registro di tutti i Digital Twin attivi nell'ecosistema. Tale registro è indicizzato per URI dei DT e mantiene tutte le informazioni generali del DT: l'identificativo, i riferimenti all'owner, i tag di classificazione e le informazioni circa il *physical asset* di riferimento.

Il Registry Manager è al tempo stesso un *listener* del DTM, al fine di ricevere la notifica si inserimento dei DT, e delle due interfacce, interna ed esterna, per ricevere le richieste relative alla funzionalità di *discovery*.

Quando riceve la notifica di *stop* di un DT, il Registry elimina il suo riferimento dal registro.

Graph Engine

L'implementazione di questo componente non è stata totalmente affrontata in questo progetto di tesi, ma l'architettura implementata è assolutamente predisposta alla sua integrazione. Si è, infatti, realizzata l'interfaccia che il componente dovrà implementare per poter elaborare le richieste relative alle query e le notifiche da parte dei DT della creazione o eliminazione di un'istanza di una relazione.

Una sua pseudo-implementazione è rappresentata dal Visualizer Graph Manager realizzato per gestire le richieste provenienti dal WoDT Platform Visualizer. Questo componente, infatti, implementa l'interfaccia descritta, al fine di poter tenere traccia dei Digital Twin registrati e delle relazioni istanziate. In questo modo, può inviare al visualizzatore della piattaforma le informazioni aggiornate circa lo stato generale dell'ecosistema.

Nello specifico, il Visualizer Graph Manager mantiene una lista dei Digital Twin registrati e una mappa delle relazioni stabilite. Ogni istanza di relazione è identificata da una chiave ed è caratterizzata dall'URI del DT che l'ha stabilita, dall'URI del DT target e da un'etichetta che riporta il nome della relazione di riferimento.

Platform Visualizer Interface

Al fine di integrare alla piattaforma un client Web adibito alla visualizzazione dello stato dell'ecosistema e dei singoli DT, si è aggiunta all'architettura, precedentemente progettata, un'ulteriore interfaccia di comunicazione, denominata Platform Visualizer Interface.

Anche in questo caso, l'interfaccia è stata realizzata attraverso un server web, non poteva essere deciso altrimenti data la necessità di interagire con un'applicazione web.

La Platform Visualizer Interface è stata implementata per consentire al visualizzatore della piattaforma di osservare in tempo reale lo stato dell'ecosistema, cioè di conoscere l'insieme dei Digital Twin attivi e quali relazioni sussistono tra di essi. Per questo motivo, l'interfaccia utilizza esclusivamente il protocollo WebSocket per poter aggiornare il visualizzatore ogniqualvolta venga creato o eliminato un DT, o istanziata o eliminata una relazione.

5.2.2 Implementazione del WoDT Platform Visualizer

Il WoDT Platform Visualizer è stato implementato con lo scopo di realizzare un'applicazione web che consentisse di visualizzare il grafo dei DT e delle relazioni attive e, al tempo stesso, permettesse di monitorare lo stato di ciascun DT.

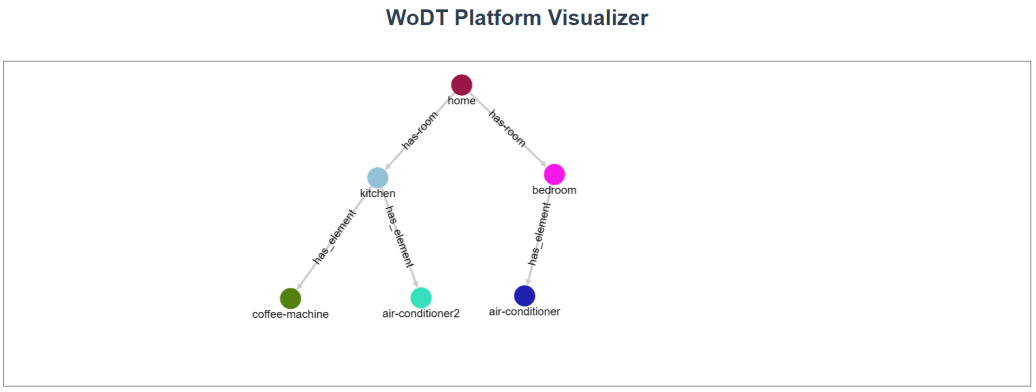
Relativamente alle tecnologie impiegate per lo sviluppo del WoDT Platform Visualizer, si è deciso di utilizzare il framework Javascript *Vue.js* e la libreria di componenti *PrimeVue*. Per la gestione grafica del grafo rappresentante l'ecosistema si è, invece, adottata la libreria *Cytoscape.js*.

Il visualizzatore è un osservatore dell'ecosistema gestito da un particolare WoDT Platform Node, per questo motivo si interfaccia attraverso il protocollo WebSocket sia con la Platform Visualizer Interface che con i singoli DT. In particolare, attraverso la piattaforma recupera le informazioni generali re-

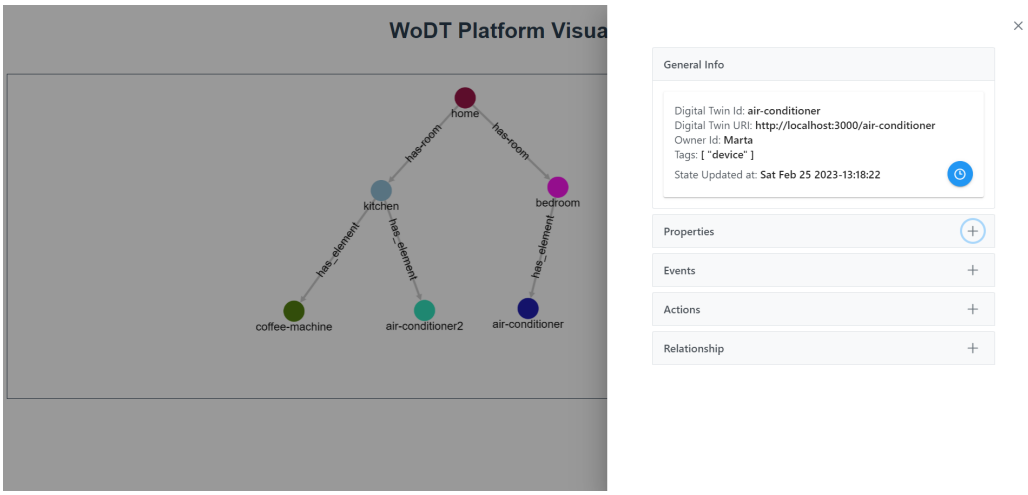
lative al grafo, cioè quali sono i DT registrati e quali sono le relazioni che sussistono tra di essi, mentre, sfruttando i Digital Twin Descriptor dei vari DT presenti nell'ecosistema, implementa un sistema di monitoraggio dello stato di ognuno.

Grazie all'utilizzo delle WebSocket l'applicazione sviluppata consente di visualizzare in tempo reale la creazione e la rimozione di ogni DT e relazione, consentendo, quindi, di monitorare l'evoluzione dell'ecosistema. Inoltre, traccia lo stato di ciascun Digital Twin, consentendo di visualizzare come si sono modificate nel tempo le proprietà e le relazioni, ma anche quando e quali eventi si sono realizzati.

In figura 5.5 sono riportate le due interfacce attualmente implementate del WoDT Platform Visualizer. I principi seguiti per la loro realizzazione sono minimalità e funzionalità, dunque sono presenti unicamente gli elementi funzionali a rendere l'interfaccia efficace.



(a) Interfaccia che mostra il grafo dei DT attualmente presenti nell'ecosistema *Smart Home*



(b) Interfaccia che mostra i dettagli dello stato di un DT

Figura 5.5: Interfacce WoDT Platform Visualizer

5.3 Esempio d'uso

Nel presente paragrafo sono presentate e descritte le operazioni per realizzare, interagire e inserire un Digital Twin in un ecosistema.

Il dominio considerato è quello di una *Smart Home*, in cui vengono modellati attraverso dei DT: la casa stessa (*Home* DT), la camera da letto (*Bedroom* DT) e la cucina (*Kitchen* DT). Inoltre, in ciascuna delle stanze sono presenti

dei *device* anch'essi associati a dei DT. Nello specifico, nella camera è presente un condizionatore (*Air Conditioner* DT), mentre, nella cucina è presente una macchina del caffè (*Coffee Machine* DT) e un altro condizionatore (*Air Conditioner* DT 2). Il DT preso in considerazione in questo esempio d'uso della libreria estesa e della piattaforma è l' *Air Conditioner* DT.

Il condizionatore fisico, installato nella camera da letto, può operare sia per riscaldare che per raffreddare la camera. Quando viene acceso, il condizionatore decide in quale modalità eseguire in base alla temperatura media a cui è impostato e alla temperatura registrata nella stanza. Oltre alla normale interazione manuale, il condizionatore è in grado di comunicare attraverso il protocollo MQTT e presenta i seguenti *topic* per inviare e ricevere messaggi:

- `air-conditioner/switch`: topic utilizzato per inviare messaggi circa l'accensione o lo spegnimento del dispositivo
- `air-conditioner/current-temp`: topic su cui sono inviati i messaggi relativi alla temperatura attualmente registrata nella stanza in cui è installato il condizionatore
- `air-conditioner/average-temp`: topic sul quale vengono inviati i messaggi relativi alla variazione della temperatura media che il condizionatore deve raggiungere.
- `air-conditioner/functioning`: topic utilizzato per inviare gli eventi che notificano una cambiamento nella modalità operativa del condizionatore.
- `air-conditioner/set-temperature`: topic su cui il condizionatore si sottoscrive per ricevere da remoto la temperatura media da raggiungere.
- `air-conditioner/set-switch`: topic sul quale il condizionatore può ricevere i comandi di accensione e spegnimento.

In questo esempio il dispositivo fisico viene simulato attraverso un apposito programma Java che emula il comportamento appena descritto ed è in grado di pubblicare e ricevere messaggi attraverso il protocollo MQTT.

Il Digital Twin considerato rappresenta il condizionatore fisico attraverso il seguente modello M:

- **Proprietà**
  - `switch`: modella lo stato dell'interruttore del condizionatore, cioè se è acceso o spento.
  - `current-temp`: rappresenta la temperatura registrata attualmente dal condizionatore.
  - `average-temp`: riporta la temperatura a cui è impostato il condizionatore, cioè la temperatura media che deve avere la stanza in cui è posto.
- **Eventi**
  - `functioning`: evento generato per segnalare la modalità di funzionamento del condizionatore, ad esempio se sta riscaldando o raffreddando la stanza.
- **Azioni**
  - `set-switch`: azione che consente di accendere o spegnere il condizionatore fisico.
  - `set-temperature`: azione per impostare la temperatura media che deve raggiungere la stanza.

Data la possibilità del condizionatore fisico di comunicare attraverso il protocollo MQTT, il relativo DT dovrà dotarsi di un MQTT Physical Adapter per supportare il processo di *shadowing*. Nel listing 2 viene riportato il codice che consente di configurare l'*adapter* considerando i *topic* definiti dal condizionatore fisico.

```
public class AirConditionerPhysicalAdapter extends MqttPhysicalAdapter {

    public final static String TOPIC_PREFIX = "air-conditioner/";
    public final static String SWITCH_PROPERTY_KEY = "switch";
    public final static String TEMPERATURE_PROPERTY_KEY = "current-temp";
    public final static String AVERAGE_TEMP_PROPERTY_KEY = "average-temp";
    public final static String FUNCTIONING_EVENT_KEY = "functioning";
    public final static String SET_TEMP_ACTION_KEY = "set-temperature";
    public final static String SET_SWICTH_ACTION_KEY = "set-switch";

    public AirConditionerPhysicalAdapter() {
        super("air-conditioner-physical-adapter", loadConfiguration());
    }

    private static MqttPhysicalAdapterConfiguration loadConfiguration() {
        return MqttPhysicalAdapterConfiguration.builder("127.0.0.1",
                                    1883,
                                    "mqtt.physical.adapter")
          .addPhysicalAssetPropertyAndTopic(SWITCH_PROPERTY_KEY,
                                    "OFF",
                                    TOPIC_PREFIX + SWITCH_PROPERTY_KEY,
                                    Function.identity())
          .addPhysicalAssetPropertyAndTopic(AVERAGE_TEMP_PROPERTY_KEY,
                                    0.0,
                                    TOPIC_PREFIX + AVERAGE_TEMP_PROPERTY_KEY,
                                    Double::parseDouble)
          .addPhysicalAssetPropertyAndTopic(TEMPERATURE_PROPERTY_KEY,
                                    0.0,
                                    TOPIC_PREFIX + TEMPERATURE_PROPERTY_KEY,
                                    Double::parseDouble)
          .addPhysicalAssetEventAndTopic(FUNCTIONING_EVENT_KEY,
                                    "event",
                                    TOPIC_PREFIX + FUNCTIONING_EVENT_KEY,
                                    Function.identity())
          .addPhysicalAssetActionAndTopic(SET_SWICTH_ACTION_KEY,
                                    "command",
                                    "text/plain",
                                    TOPIC_PREFIX + SET_SWICTH_ACTION_KEY,
                                    Function.identity())
          .addPhysicalAssetActionAndTopic(SET_TEMP_ACTION_KEY,
                                    "command",
                                    "text/plain",
                                    TOPIC_PREFIX + SET_TEMP_ACTION_KEY,
                                    Function.identity())
          .build();
    }
}
```

Listing 2: Definizione di un MQTT Physical Adapter per il DT di un condizionatore

A seguito della connessione all'MQTT Broker assegnato, l'MQTT Physical Adapter produrrà la sua Physical Asset Description (PAD), cioè l'oggetto che definisce l'insieme delle proprietà, azioni, eventi e relazioni che l'entità fisica espone attraverso tale protocollo.

Oltre al *physical adapter*, per garantire la sincronizzazione tra l'entità fisica e la replica digitale, è necessario implementare una *WoDT Shadowing Function*. Attraverso l'implementazione della funzione di *shadowing*, il progettista determina come le diverse componenti delle varie PAD sono mappate sul modello M del DT. Nel listato di codice 3 è riportato un esempio di implementazione del metodo `onDigitalTwinBoundEvent` della *WoDT Shadowing Function*, quest'ultimo è eseguito quando il Digital Twin transita nello stato *Bound*, cioè quando tutti i Physical Adapter hanno stabilito una connessione con l'entità fisica e prodotto la propria PAD. Nell'esempio si mostra il caso in cui a ciascuna proprietà, evento, azione e relazione definita in un PAD corrisponda un elemento dello stato del DT. Implementando tale metodo occorre definire anche quali componenti si vuole osservare, in modo tale da ricevere i relativi eventi e_{PA}.

Nel listato di codice 4 viene riportata una possibile implementazione del metodo `onPhysicalAssetPropertyVariation`. Quest'ultimo consente alla funzione di *shadowing* di definire come gestire un e_{PA} associato alla modifica di una proprietà. In questo caso si è scelto di modellare il comportamento base, cioè la modifica viene direttamente riflessa nello stato del Digital Twin ma un progettista può definire un comportamento arbitrariamente complesso.

Definita la funzione di *shadowing* e l'*adapter*, è possibile implementare il DT istanziando la sua *WLDT Engine* e associandole i diversi componenti. Nel listing 5 è riportato il codice necessario per completare l'implementazione del *Air Conditioner* Digital Twin.

```
public void onDigitalTwinBoundEvent(
              Map<String, PhysicalAssetDescription> physicalAssetDescriptionMap
) {
    adaptersPhysicalAssetDescriptionMap.forEach((id, pad) -> {
        pad.getProperties().forEach(p -> {
            if(!this.digitalTwinState.containsProperty(p.getKey()))
                this.digitalTwinState.createProperty(
                    new DigitalTwinStateProperty<>(p.getKey(),p.getInitialValue())
                );
        });
        pad.getActions().forEach(a -> {
            if(!this.digitalTwinState.containsAction(a.getKey()))
                this.digitalTwinState.enableAction(
                    new DigitalTwinStateAction(a.getKey(), a.getType(), a.getContentType())
                );
        });
        pad.getEvents().forEach(e -> {
            if(!this.digitalTwinState.containsEvent(e.getKey()))
                this.digitalTwinState.registerEvent(
                    new DigitalTwinStateEvent(e.getKey(), e.getType())
                );
        });
        pad.getRelationships().forEach(r -> {
            if(!this.digitalTwinState.containsRelationship(r.getName()))
                this.digitalTwinState.createRelationship(
                    new DigitalTwinStateRelationship<>(r.getName(), "test-type")
                );
        });
        notifyShadowingSync();
    });

    this.observePhysicalAssetProperties(getComponentList(physicalAssetDescriptionMap,
            pad -> pad.getProperties().stream()));

    this.observePhysicalAssetEvents(getComponentList(physicalAssetDescriptionMap,
            pad -> pad.getEvents().stream()));

    this.observePhysicalAssetRelationships(getComponentList(physicalAssetDescriptionMap,
            pad -> pad.getRelationships().stream()));

    observeDigitalActionEvents();
}
```

Listing 3: Esempio di implementazione del metodo che notifica il binding del DT con l'entità fisica

```
@Override
protected void onPhysicalAssetPropertyVariation(
                    PhysicalAssetPropertyWldtEvent<?> physicalPropertyEventMessage
) {
    try {
        this.digitalTwinState.updateProperty(
                new DigitalTwinStateProperty<>(
                        physicalPropertyEventMessage.getPhysicalPropertyId(),
                        physicalPropertyEventMessage.getBody()
                )
        );
    } catch (WldtDigitalTwinStatePropertyException |
          WldtDigitalTwinStatePropertyBadRequestException |
          WldtDigitalTwinStatePropertyNotFoundException |
          WldtDigitalTwinStateException e) {
        e.printStackTrace();
    }
}
```

Listing 4: Esempio di implementazione del metodo che notifica alla funzione di *shadowing* l'aggiornamento di un proprietà dell'*asset fisico*

```
public class AirConditionerDT {
    public static void main(String[] args) {
        WldtEngine dt = new WldtEngine(
                new WoDTdefaultShadowingFunction(),
                DigitalTwinEnvironment.getInstance().getDigitalTwinId()
        );
        dt.addPhysicalAdapter(new AirConditionerPhysicalAdapter());

        WoDTDigitalAdapterConfiguration config = new WoDTDigitalAdapterConfiguration(
                "air-conditioner-digital-adapter",
                DigitalTwinEnvironment.getInstance().getDigitalTwinHost(),
                DigitalTwinEnvironment.getInstance().getDigitalTwinPort()
        );
        dt.addDigitalAdapter(new WoDTDigitalAdapter(config));

        dt.startLifeCycle();
    }
}
```

Listing 5: Esempio di implementazione della WLDT Engine del DT del condizionatore

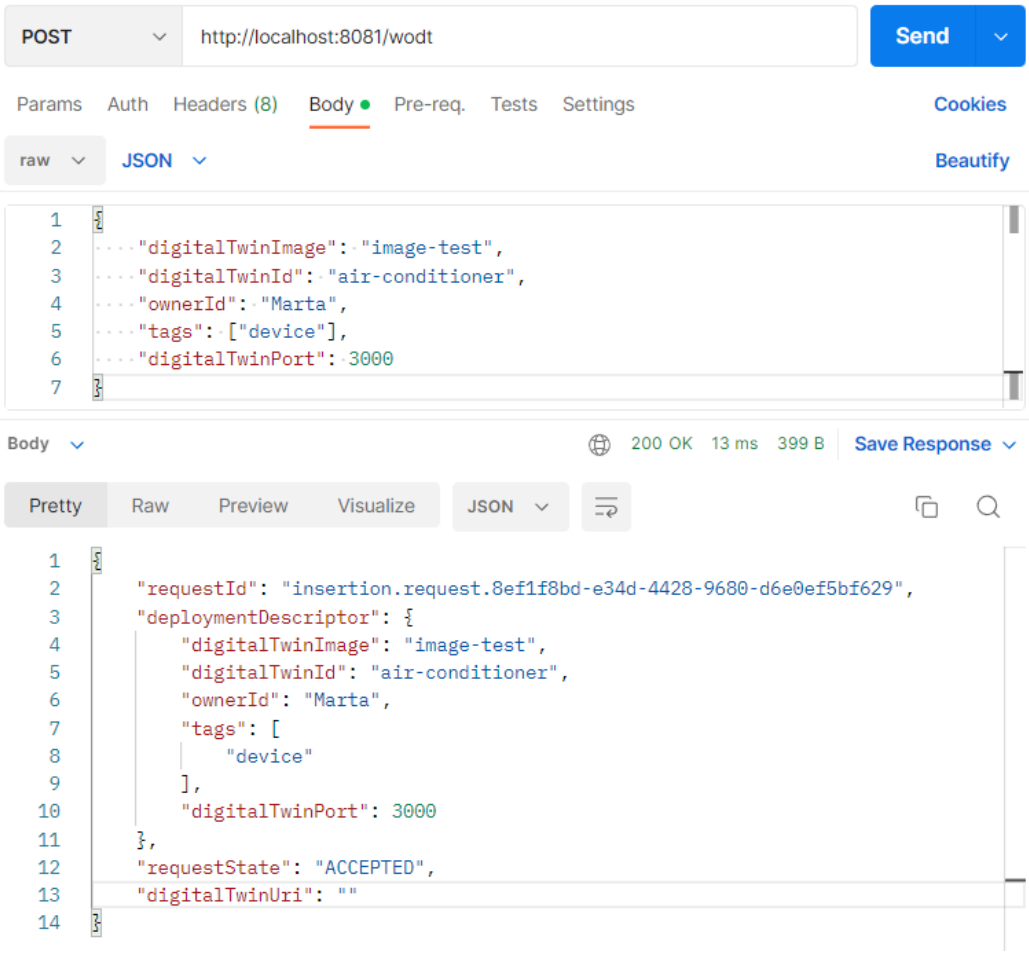


Figura 5.6: Esempio di richiesta di inserimento per l'Air Conditioner DT e relativa risposta

In seguito all'esecuzione del WoDT Platform Node è possibile richiedere l'inserimento nell'ecosistema di un Digital Twin attraverso un'apposita richiesta HTTP. In figura 5.6, sono riportate la richiesta POST effettuata per inserire l'*Air Conditioner DT* nell'ecosisitema della *Smart Home*, e la risposta ottenuta, cioè la Insertion Request associata al DT. Come descritto precedentemente, eseguendo un *polling* sulla risorsa ricevuta in risposta è possibile determinare quando il DT è stato correttamente avviato e incluso nell'ecosistema. In figura 5.7 è mostrata la richiesta da effettuare per eseguire il *polling* e la risposta che si ottiene quando l'inserimento è andato a buon fine.

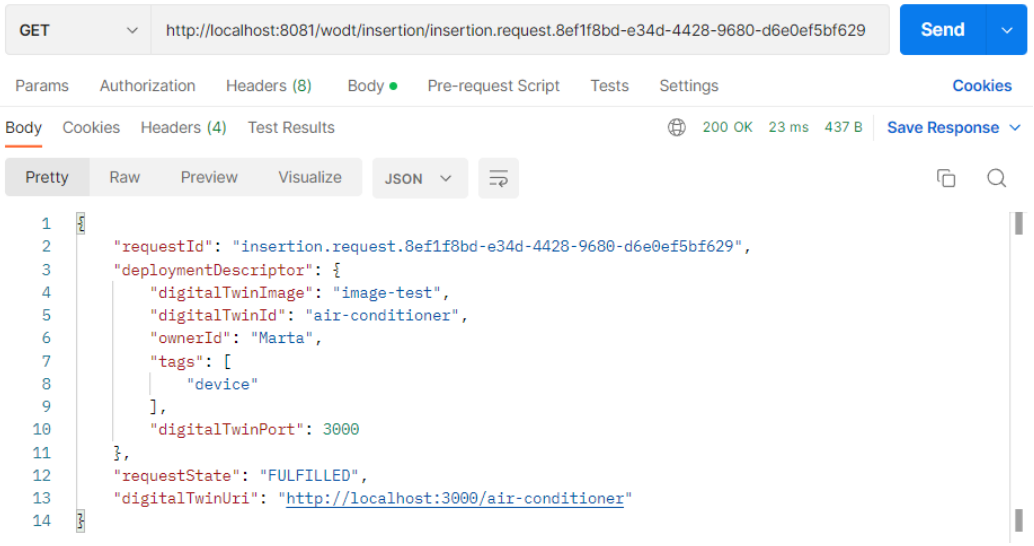


Figura 5.7: Esempio di richiesta per eseguire il polling sullo stato della richiesta di inserimento

Effettuando una richiesta GET all'indirizzo presente in corrispondenza del campo `digitalTwinUri` è possibile richiedere al DT stesso il proprio descrittore. Processando il *Digital Twin Descriptor* un'applicazione può determinare come interagire con il Digital Twin. In figura 5.8 è riportata una porzione del DT-Descriptor dell'*Air Conditioner* DT, nella quale sono mostrati i form per interagire con la proprietà `current-temp`.

Se si esegue la richiesta associata all'operazione `observeproperty` è possibile osservare tale proprietà. In figura 5.9 sono riportati i messaggi ricevuti osservando la proprietà mentre il condizionatore fisico sta riscaldando la stanza.

Oltre ad interagire con il singolo DT presente nell'ecosistema, un'applicazione può richiedere alcuni servizi al *WoDT Platform Node* stesso. In particolare, in figura 5.10 è riportato un esempio di richiesta e risposta relative al servizio di *yellow-page* offerto dal Registry Manager. Nell'esempio si simula la richiesta da parte di un'applicazione di ottenere gli indirizzi di tutti i DT, presenti nell'ecosistema, classificati come "device". La classificazione avviene da parte dell'*owner* del singolo DT al momento della richiesta di inserimento, specificando i relativi `tags` (figura 5.6).

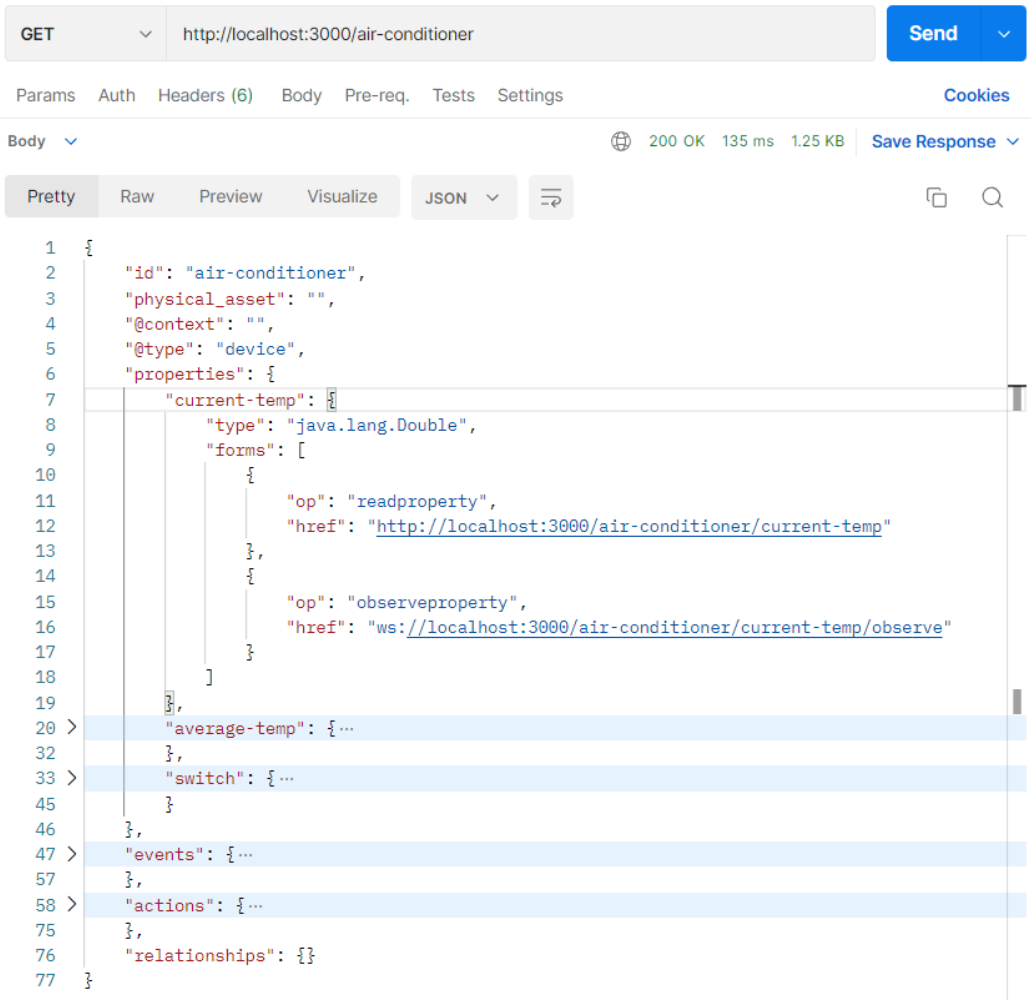


Figura 5.8: Richiesta per ottenere il Digital Twin Descriptor e relativa risposta

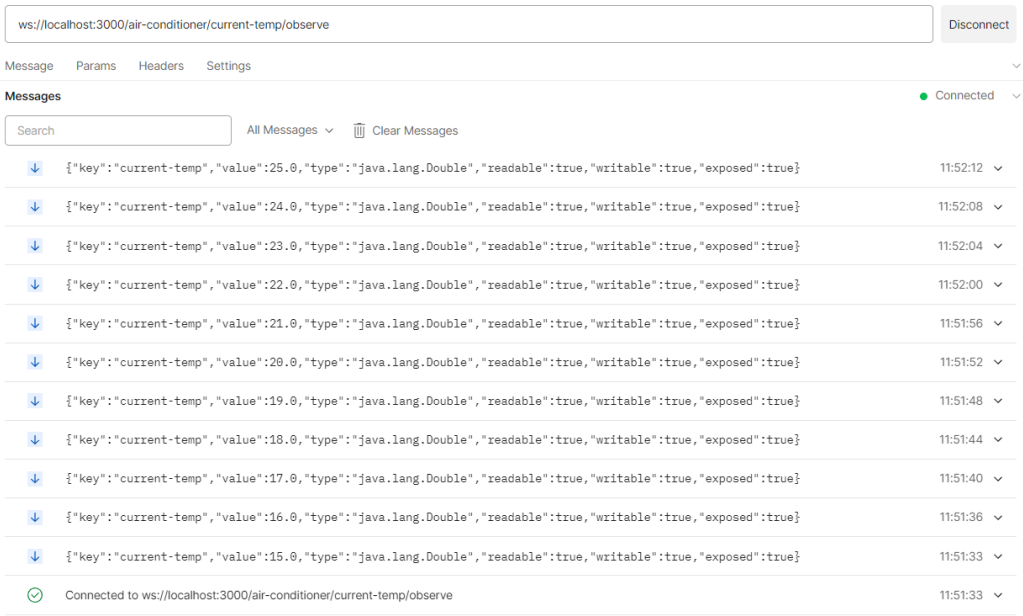


Figura 5.9: Connessione alla WebSocket adibita ad osservare una proprietà

GET http://localhost:8081/wodt/digitalTwins/tags/device Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Body 200 OK 15 ms 659 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "digitalTwinURI": {
            "uri": "http://localhost:3004/air-conditioner2"
        },
        "digitalTwinId": "air-conditioner2",
        "ownerId": "Marta",
        "tags": [
            "device"
        ],
        "registrationTimestamp": 1678014599745
    },
    {
        "digitalTwinURI": {
            "uri": "http://localhost:3005/coffee-machine"
        },
        "digitalTwinId": "coffee-machine",
        "ownerId": "Marta",
        "tags": [
            "device"
        ],
        "registrationTimestamp": 1678014603019
    },
    {
        "digitalTwinURI": {
            "uri": "http://localhost:3000/air-conditioner"
        },
        "digitalTwinId": "air-conditioner",
        "ownerId": "Marta",
        "tags": [
            "device"
        ],
        "registrationTimestamp": 1678014592892
    }
]
```


Figura 5.10: Esempio di richiesta per il servizio di ricerca *yellow-page* e relativa risposta per l'ecosistema della *Smart Home*

Conclusioni

Questo progetto di tesi nasce dalla volontà di realizzare la prima implementazione di una piattaforma che supporti Web of Digital Twins [2]. La visione del paradigma Digital Twin, ideata in WoDT, consente di rappresentare sistemi complessi e su larga scala attraverso ecosistemi aperti, dinamici e distribuiti. L'eterogeneità nella modellazione e nell'implementazione dei DT presente allo stato dell'arte ha determinato la formazione di *silos* software che non consentono di sfruttare a pieno le potenzialità del paradigma. WoDT mira a superare questa visione ristretta e focalizzata su un singolo dominio applicativo al fine di generare un sistema interoperabile, fruibile attraverso i protocolli propri del Web, in cui i singoli DT possono essere utilizzati *as-a-service* da infrastrutture di domini differenti.

Nel contesto di questa tesi, sono stati progettati e implementati tre dei principali elementi che consentono di realizzare un Web of Digital Twins: un framework per l'implementazione di DT in ottica WoDT, il concetto di *Digital Twin Descriptor* e la *WoDT Platform*.

La libreria White Label Digital Twin è stata estesa introducendo il concetto di relazione tra DT e implementando il supporto per la comunicazione con il livello applicativo e con la piattaforma che gestisce l'ecosistema. In questo modo, si è realizzato un framework che consente di definire i DT come delle singole entità attive fruibili in modo indipendente e interoperabile da applicazioni *cross-domain*. Nello specifico, ciascun DT rappresenta un servizio indipendente che modella la relativa entità fisica attraverso proprietà, eventi, azioni e relazioni. La possibilità di replicare a livello digitale le connessioni presenti tra le entità fisiche consente di riflettere a pieno il mondo reale, nel quale, infatti, le diverse risorse possono muoversi e interagire in contesti diversi allo stesso tempo.

La modellazione del concetto di *Digital Twin Descriptor* è sicuramente

uno dei contributi principali di questo elaborato. Il *DT-Descriptor* rappresenta, infatti, l'elemento fondamentale per garantire interoperabilità, in quanto consente al singolo Digital Twin di esporre la propria API alle applicazioni interessate ad interagirvi. Attraverso il descrittore, inoltre, le applicazioni possono contestualizzare a livello semantico il DT, cioè possono determinare che cosa rappresenti nel dominio in cui è collocato.

La piattaforma realizzata è, invece, il componente che abilita alla possibilità di definire e gestire un ecosistema di *Digital Twin*. In questo modo, è possibile realizzare un layer omogeneo attraverso il quale sopperire alla frammentazione e all'eterogeneità che caratterizza il livello fisico. La piattaforma supporta la gestione di un insieme di Digital Twin, i quali possono essere in esecuzione in punti diversi della rete e possono anche appartenere ad organizzazioni differenti. La scelta di implementare la WoDT Platform utilizzando i protocolli tipici del Web, consente, inoltre, di integrarla anche con Digital Twin realizzati attraverso tecnologie diverse dalla libreria estesa.

Il prototipo della WoDT Platform implementato, nonostante non possa essere considerato un prodotto completo, abilita, già in questa versione, alla definizione di ecosistemi, in cui i DT sono in grado di instaurare relazioni e possono essere fruiti dalle diverse applicazioni. Un aspetto da integrare, in un possibile sviluppo futuro, è l'avvio automatico dei DT da parte del Digital Twin Manager. Ciascun DT, al pari di un microservizio, può essere distribuito attraverso un container Docker che dunque, potrà essere adeguatamente eseguito dal DTM nel nodo di esecuzione assegnato al DT stesso.

Rispetto ai requisiti individuati, il tema dell'osservazione del grafo e delle query su di esso, o su una sua porzione, è sicuramente quello meno indagato in questo progetto di tesi e che si pone come un interessante sviluppo futuro. Le architetture della piattaforma e dei singoli DT sono entrambe predisposte all'integrazione dei moduli necessari per realizzare tale requisito. L'obiettivo sarebbe quello di poter definire, attraverso un apposito linguaggio, il concetto semantico che si vuole indagare o osservare considerando l'insieme dei DT e le relazioni che intercorrono tra di essi. Ad esempio, nel caso della *Smart Home*, una possibile query sarebbe *"ottenere tutte le stanze del primo piano che hanno la luce accesa*".

Il *WoDT Platform Visualizer* implementato, oltre a fornire un riscontro visuale dell'ecosistema definito, consente di visualizzare l'evoluzione nel

tempo dello stato di un Digital Twin. Integrando al modello dei DT il tracciamento degli eventi che determinano i cambiamenti di stato, in futuro, non solo si avrebbe la possibilità di indagare il presente ma anche di ricostruire e interrogare il passato. Ad esempio, potrebbe essere possibile visualizzare quale fosse lo stato di una determinata entità del sistema prima di una sua rottura o problematica, al fine di attuare le adeguate azioni di *recovery*.

La piattaforma progettata, il Digital Twin Descriptor e la nuova versione della libreria WLDT sono un primo passo concreto per la realizzazione di un Web of Digital Twins, il quale abilita i DT a cooperare al fine di realizzare funzionalità impossibili se si considerasse unicamente la realtà fisica o la visione a *silos* presente attualmente allo stato dell'arte.

Bibliografia

[1] Roberto Minerva, Gyu Myoung Lee, and Noël Crespi. Digital twin in the iot context: A survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE*, 108(10):1785–1824, 2020. doi: 10.1109/JPROC.2020.2998530.

[2] Alessandro Ricci, Angelo Croatti, Stefano Mariani, Sara Montagna, and Marco Picone. Web of digital twins. *ACM Transactions on Internet Technology*, 22(4):1–30, 2022.

[3] Alessandro Ricci, Angelo Croatti, and Sara Montagna. Pervasive and connected digital twins—a vision for digital health. *IEEE Internet Computing*, 26(5):26–32, 2022. doi: 10.1109/MIC.2021.3052039.

[4] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. *Transdisciplinary perspectives on complex systems: New findings and approaches*, pages 85–113, 2017.

[5] Michael Grieves. Digital twin: Manufacturing excellence through virtual factory replication. 2014.

[6] Barbara Rita Barricelli, Elena Casiraghi, and Daniela Fogli. A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE Access*, 7:167653–167671, 2019. doi: 10.1109/ACCESS.2019.2953499.

[7] Roland Rosen, Georg Von Wichert, George Lo, and Kurt D Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. *Ifac-Papersonline*, 48(3):567–572, 2015.

[8] Stephen Ferguson. Apollo 13: The first digital twin. https://blogs.sw.siemens.com/simcenter/apollo-13-the-first-digital-twin/, 2020. Accessed: gennaio 2023.

[9] Shaft Mike, Conroy Mike, Doyle Rich, Glaessgen Ed, Kemp Chris, LeMoigne Jacqueline, and Wang Lui. Modeling, simulation, information technology & processing roadmap. technology area 11, November 2010. URL https://www.nasa.gov/pdf/501321main_TA11-MSITP-DRAFT-Nov2010-A1.pdf.

[10] Fei Tao, He Zhang, Ang Liu, and A. Y. C. Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4): 2405–2415, 2019. doi: 10.1109/TII.2018.2873186.

[11] Gartner. Top 10 technology trends 2017. https://www.gartner.com/smarterwithgartner/gartners-top-10-technology-trends-2017, 2017. Accessed: gennaio 2023.

[12] Gartner. Top 10 strategic technology trends for 2018. https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2018, 2018. Accessed: gennaio 2023.

[13] Gartner. Top 10 strategic technology trends for 2019. https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2019, 2019. Accessed: gennaio 2023.

[14] 5 trends drive the gartner hype cycle for emerging technologies. https://www.gartner.com/smarterwithgartner/5-trends-drive-the-gartner-hype-cycle-for-emerging-technologies-2020, 2020. Accessed: gennaio 2023.

[15] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. *Transdisciplinary perspectives on complex systems: New findings and approaches*, pages 85–113, 2017.

[16] Maulshree Singh, Evert Fuenmayor, Eoin P Hinchy, Yuansong Qiao, Niall Murray, and Declan Devine. Digital twin: Origin to future. *Applied System Innovation*, 4(2):36, 2021.

[17] Fei Tao, Bin Xiao, Qinglin Qi, Jiangfeng Cheng, and Ping Ji. Digital twin modeling. *Journal of Manufacturing Systems*, 64:372–389, 2022.

[18] Qinglin Qi, Dongming Zhao, T. Warren Liao, and Fei Tao. Modeling of cyber-physical systems and digital twin based on edge computing, fog computing and cloud computing towards smart manufacturing. Volume 1: Additive Manufacturing; Bio and Sustainable Manufacturing, 2018. doi: 10.1115/MSEC2018-6435. URL `https://doi.org/10.1115/MSEC2018-6435`.

[19] Roberto Minerva and Noël Crespi. Digital twins: Properties, software frameworks, and application scenarios. *IT Professional*, 23(1):51–55, 2021.

[20] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022, 2018. doi: https://doi.org/10.1016/j.ifacol.2018.08.474. URL `https://www.sciencedirect.com/science/article/pii/S2405896318316021`.

[21] Angelo Croatti, Matteo Gabellini, Sara Montagna, and Alessandro Ricci. On the integration of agents and digital twins in healthcare. *Journal of Medical Systems*, 44:161, 08 2020. doi: 10.1007/s10916-020-01623-5.

[22] Roberto Saracco. Digital twins: Bridging physical space and cyberspace. *Computer*, 52(12):58–64, 2019. doi: 10.1109/MC.2019.2942803.

[23] Bergthor Bjornsson, Carl Borrebaeck, Nils Elander, Thomas Gasslander, Danuta Gawel, Mika Gustafsson, Rebecka Jörnsten, Eun Jung Lee, Xinxiu Li, Sandra Lilja, David Martínez-Enguita, Andreas Matussek, Per Sandström, Samuel Schäfer, Margaretha Stenmarker, X. Sun, Oleg Sysoev, Huan Zhang, and Mikael Benson. Digital twins to personalize medicine. *Genome Medicine*, 12, 12 2019. doi: 10.1186/s13073-019-0701-3.

[24] Tolga Erol, Arif Furkan Mendi, and Dilara Doğan. The digital twin revolution in healthcare. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–7, 2020. doi: 10.1109/ISMSIT50672.2020.9255249.

[25] Neda Mohammadi and John E. Taylor. Smart city digital twins. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–5, 2017. doi: 10.1109/SSCI.2017.8285439.

[26] Stefan Mihai, Mahnoor Yaqoob, Dang V. Hung, William Davis, Praveer Towakel, Mohsin Raza, Mehmet Karamanoglu, Balbir Barn, Dattaprasad Shetve, Raja V. Prasad, Hrishikesh Venkataraman, Ramona Trestian, and Huan X. Nguyen. Digital twins: A survey on enabling technologies, challenges, trends and future prospects. *IEEE Communications Surveys & Tutorials*, 24(4):2255–2291, 2022. doi: 10.1109/COMST.2022.3208773.

[27] Fei Tao and Qinglin Qi. Make more digital twins. *Nature*, 573:490–491, 2019.

[28] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52, 2020. ISSN 1755-5817. doi: https://doi.org/10.1016/j.cirpj.2020.02.002.

[29] Digital twin consortium. `https://www.digitaltwinconsortium.org/`. Accessed: gennaio 2023.

[30] Standard iso 23247-1 digital twin framework for manufactoring. `https://www.iso.org/standard/75066.html`. Accessed: gennaio 2023.

[31] Marco Picone, Marco Mamei, and Franco Zambonelli. A flexible and modular architecture for edge digital twin: Implementation and evaluation. *ACM Transactions on Internet of Things*, 2022.

[32] Center for Digital Built Britan. The gemini principles. `https://www.cdbb.cam.ac.uk/system/files/documents/TheGeminiPrinciples.pdf`, 2018. Accessed: gennaio 2023.

[33] David Gelernter. *Mirror worlds: Or the day software puts the universe in a shoebox... How it will happen and what it will mean.* Oxford University Press, 1993.

Ringraziamenti

Un sentito ringraziamento al Prof. Alessandro Ricci, al Prof. Marco Picone e Dott. Samuele Burattini per avermi guidato e supportato nella realizzazione di questa tesi. Lavorare con voi è stato un piacere e un onore.

Un altro enorme ringraziamento va ai miei genitori per aver sempre creduto in me e avermi sostenuto in questo lungo percorso.

Non posso poi non ringraziare i miei colleghi, con una menzione speciale a Martina, Alessandro, Simone e Matteo, per le risate, il sostegno e i confronti dai quali ho sempre imparato qualcosa di nuovo.

Infine, un grazie ad Alessia, per esserci sempre, a Michele, per gli infiniti passaggi e le discussioni sugli argomenti più disparati, e a tutti colori che mi sono stati vicini in questi anni. Senza tutti voi non sarebbe stato lo stesso, grazie.