Requisiti software per Kafka e data streams per Digital Twins

Autore: Daniele Greco, Matias Negro, Nicolò Zaffaroni
Standard: ISO/IEC/IEEE 29148

Data: 06/04/2025
Stato: Bozza
Versione: 1.8

Sommario

1. Introduzione

1.1 Scopo del documento

Questo documento specifica i requisiti per la progettazione e implementazione di una pipeline software per gestire lo scambio di dati tra il Physical Twin (simulazioni e sensori del paziente e del ventilatore) e il Digital Twin.

Il sistema è progettato per supportare:

- Trasmissione unidirezionale e bidirezionale di dati.
- Registrazione accurata dei messaggi in uno storage esterno.
- Scalabilità e modularità attraverso l'uso di container Docker.

1.2 Scopo del progetto

L'obiettivo del progetto è implementare una pipeline dati che permetta:

1. La trasmissione unidirezionale e bidirezionale tra Physical e Digital Twin.
2. La registrazione completa di ogni scambio di dati.
3. L'archiviazione dei dati raccolti per analisi successive.
4. L'integrazione di traduttori ZeroMQ-Kafka e REST-Kafka per convertire i messaggi tra i Twin, garantendo una comunicazione asincrona e scalabile.

1.3 Panoramica

Il documento include:

- Descrizione delle funzionalità richieste per la pipeline dati.
- Requisiti funzionali, non funzionali e di sicurezza.
- Specifiche tecniche per implementare la pipeline.

2. Descrizione generale

2.1 Prospettiva del prodotto

La pipeline funge da intermediario per la trasmissione dei dati tra:

1. Paziente ospedaliero: Genera dati relativi allo stato di salute del paziente.
2. Ventilatore polmonare: Invio di parametri operativi e ricezione di istruzioni.
3. Digital Twin: Modello digitale che, attraverso i dati a lui forniti, simula il comportamento del relativo Physical Twin.

Il sistema includerà un wrapper per consentire l'integrazione tra il middleware ZeroMQ e Kafka, facilitando la comunicazione bidirezionale tra i componenti. Il wrapper ZeroMQ-Kafka traduce i messaggi pubblicati da ZeroMQ in messaggi Kafka, garantendo interoperabilità, ed è progettato per mantenere i metadati originali (timestamp, topic, tipo di messaggio) durante la conversione.

Tutti i dati saranno scambiati in formato JSON, con il middleware che implementa il pattern pub-sub tramite ZeroMQ. I seguenti topic sono definiti per gestire lo scambio dei dati:

- PatientParameters: Dati relativi al simulatore del paziente (es. frequenza cardiaca, volume polmonare).
- VentilatorParameters: Dati relativi al ventilatore (es. pressione, flusso).

2.2 Funzionalità del prodotto

Descrizione delle funzionalità principali offerte dal sistema, evidenziando come ogni componente contribuisce al funzionamento complessivo. Le funzionalità sono orientate alla gestione dei dati tra il Physical Twin e il Digital Twin.

1. Trasmissione dati: Trasferimento in tempo reale dei dati tra generatori e Digital Twin.
2. Registrazione e logging: Ogni scambio di dati deve essere registrato con dettagli temporali.
3. Archiviazione: Salvataggio di tutti i dati in uno storage esterno.
4. Scalabilità: Possibilità di aggiungere nuovi componenti senza modificare l’architettura.

2.3 Utenti e utilizzo

Possibili utenti del sistema con i loro relativi utilizzi:

- Ricercatori: Analizzano i dati storici, inizializzano la simulazione e aggiungono simulatori.
- Sviluppatore: Crea interfacce e configura Kafka.
- Physical twins: Generano e scambiano dati.

2.4 Vincoli

Descrizione dei vincoli tecnici e operativi del progetto, tra cui i formati dati supportati, i protocolli utilizzati, e i limiti architetturali. Tali vincoli definiscono le regole entro cui il sistema deve operare.

- Formato dati: JSON per la trasmissione e archiviazione.
- Protocolli: ZeroMQ e REST per far comunicare i moduli con Kafka.

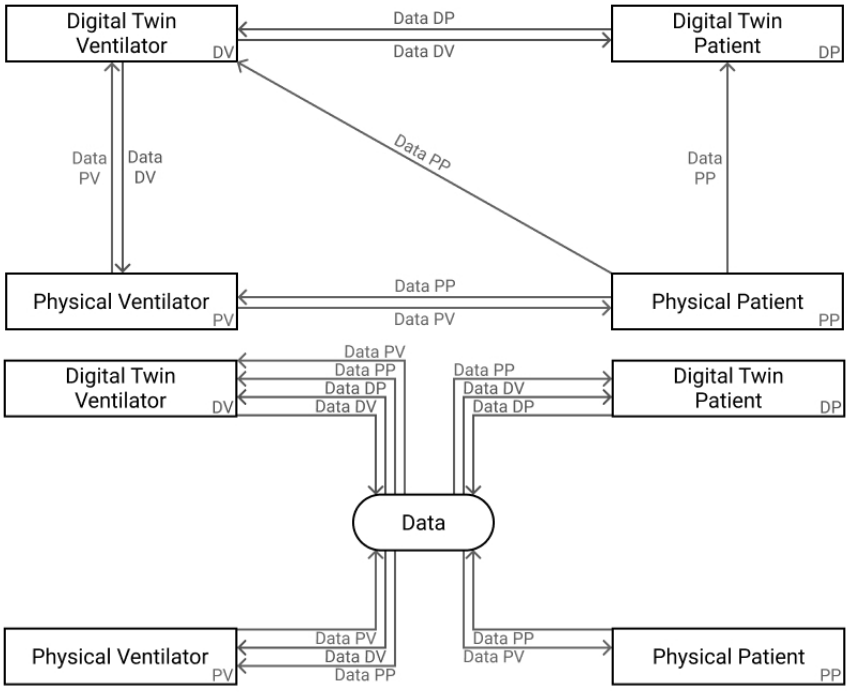
3. Requisiti

3.1 Requisiti principali

La tabella seguente riassume i requisiti principali del sistema, organizzandoli per identificatore (ID) e descrizione. Rappresenta una panoramica dei requisiti chiave che saranno soddisfatti durante lo sviluppo.

ID	Requisito
RF-1	Il sistema deve inviare messaggi JSON dai simulatori al Digital Twin.
RF-2	Ogni messaggio deve includere timestamp, origine e tipo di dato.
RF-3	I messaggi ricevuti devono essere salvati in file di log.
RF-4	Deve essere possibile aggiungere nuovi topic senza modificare l'architettura.
RF-5	Il Digital Twin deve essere in grado di ricevere e processare i dati in tempo reale dai simulatori.
RF-6	Il sistema deve essere containerizzato per garantire portabilità.

Diagrammi che illustrano lo scambio di dati tra le diverse entità della pipeline, inclusi i topic e i messaggi JSON:



3.2 Requisiti di sistema

Descrizione dei requisiti di alto livello che il sistema deve soddisfare per garantire un'operatività ottimale.

- Il sistema deve supportare comunicazioni bidirezionali tra Physical Twin e Digital Twin.
- Deve essere possibile archiviare tutti i dati generati in uno storage esterno.
- Ogni componente del sistema deve essere containerizzato per garantire la portabilità.

3.3 Requisiti degli stakeholder

Elenco dei requisiti identificati dalle parti interessate del progetto, come ricercatori e sviluppatori. Questi requisiti guidano la progettazione del sistema per soddisfare le esigenze specifiche di ciascun gruppo.

- Gli stakeholder richiedono un sistema di immagazzinamento dei dati scambiati tra i vari componenti, in modo tale da poterli tracciare e analizzare.
- I ricercatori necessitano di un sistema scalabile per analizzare i dati storici e validare i modelli predittivi.
- Gli sviluppatori richiedono un'infrastruttura modulare basata su standard aperti (JSON, ZeroMQ, Kafka) per facilitare l'integrazione e la manutenzione.

3.4 Descrizione delle componenti chiave

Descrizione nei dettagli di ogni componente del sistema, con focus su protocollo, input/output e comportamento.

La presente sezione è accompagnata da diagrammi UML che supportano la descrizione strutturale e comportamentale del sistema, fornendo una rappresentazione visuale delle interazioni tra entità, dei flussi di dati e dell’architettura.

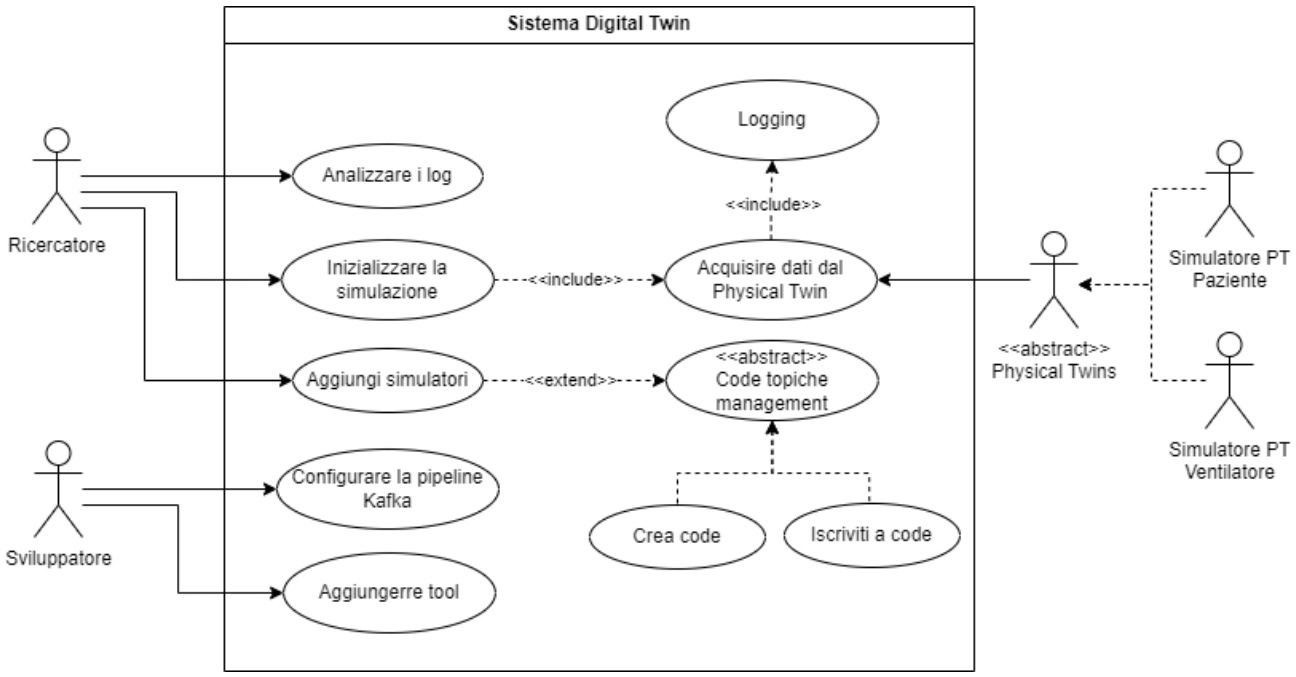
- Patient Simulator (Physical Twin)
    - Descrizione Generale: Il Patient Simulator è responsabile per la simulazione dei dati relativi al paziente, come la frequenza cardiaca, il volume polmonare e altri parametri vitali. I dati simulati sono essenziali per alimentare il Digital Twin del paziente e il ventilatore.
    - Protocollo: ZeroMQ
    - Input:
        - Formato: JSON
        - Unità di misura: Ogni dato deve includere l'unità di misura (ad esempio mL per volume, per_min per la frequenza respiratoria).
        - Frequenza nominale: Ogni 0.02 secondi.
    - Output:
        - Formato: JSON con struttura gerarchica che include il nome, il valore e l'unità di misura.

- Ventilator Software (Physical Twin)
    - Descrizione Generale: Il software del ventilatore fornisce i dati relativi al flusso d'aria e alle impostazioni del ventilatore, che influenzano direttamente i parametri del paziente.
    - Protocollo: ZeroMQ
    - Input: Stesso formato dell'output del Patient Simulator.
    - Output: Formato simile all'input del Patient Simulator, ma applicato ai parametri del ventilatore.

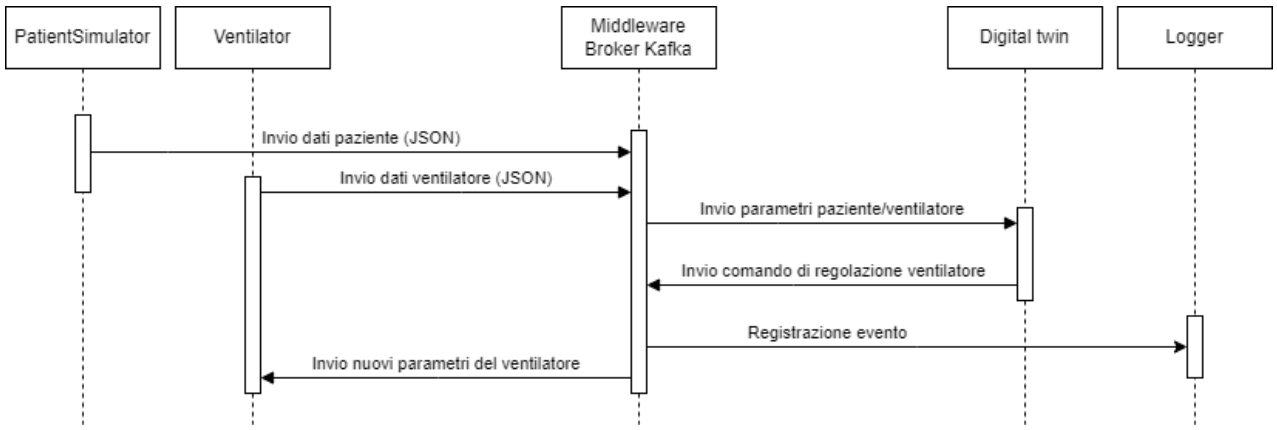
- ASM-Based Ventilator Model (Digital Twin)
    - Descrizione Generale: Il modello ASM-Based Ventilator rappresenta la simulazione del ventilatore nel Digital Twin. Utilizza il software AsmetaS@Runtime per simulare il comportamento del ventilatore basato sui dati in ingresso.
    - Input: Lista separata da virgole con i nomi delle funzioni monitorate e i relativi valori.
    - Output: Lista separata da virgole con i nomi delle funzioni di output e i relativi valori.
    - REST-Based Server: Modalità di trasmissione sincrona (richiesta/risposta) per interazioni pull tra il middleware e il modello.

- ASM-Based Patient (Digital Twin)
    - Descrizione Generale: Il Digital Twin del Paziente è il modello virtuale che replica i dati fisiologici del paziente simulato, interagendo con il ventilatore per regolare i parametri vitali.
    - Input: Lista separata da virgole con i nomi delle funzioni monitorate e i relativi valori.
    - Output: Lista separata da virgole con i nomi delle funzioni di output e i relativi valori.
    - REST-Based Server: Modalità di trasmissione sincrona, con pull per entrambe le direzioni.

- UML - Use case diagram



- UML - Sequence diagram



3.5 Tracciabilità dei requisiti

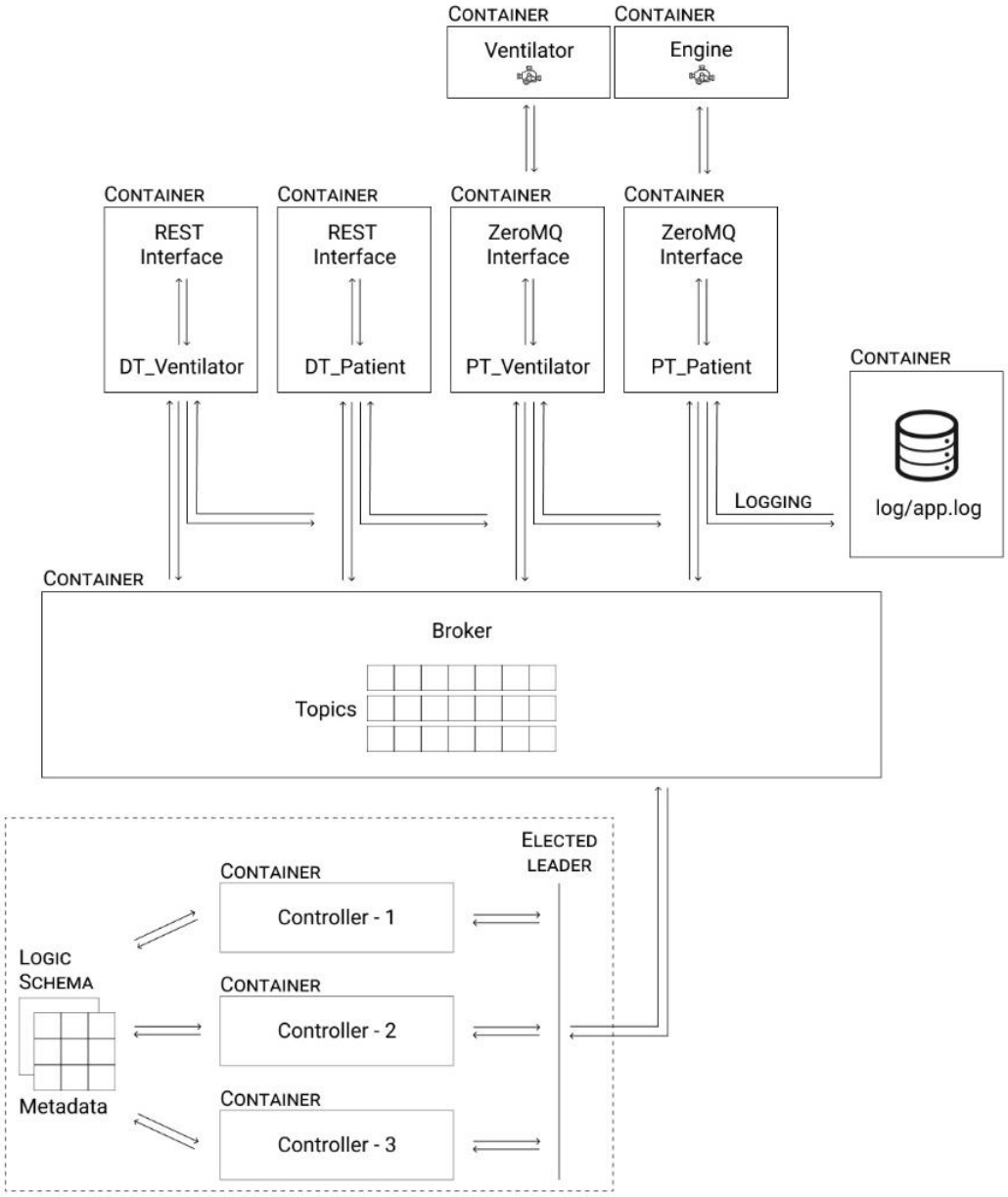
La matrice di tracciabilità dei requisiti collega i requisiti principali alle implementazioni e ai metodi di verifica. Questa tabella garantisce che ogni requisito sia soddisfatto attraverso il design e test appropriati.

ID	Stakeholder	Implementazione	Verifica
RF-1	Sviluppatori	Producer/Consumer con JSON su Pulse e BREATHE	Test di trasmissione dati real-time
RF-2	Ricercatori	Logger con timestamp JSON	Verifica integrità log
RF-3	Ricercatori	Logger scrive su log/app.log	Test accesso e consistenza dati
RF-4	Tutti	Kafka con topic configurabili	Test aggiunta dinamica topic
RF-5	Ricercatori	Digital Twin processa i dati in tempo reale	Test latenza < 50 ms
RF-6	Sviluppatori	Containerizzazione con Docker	Verifica esecuzione in container

4. Specifiche tecniche

4.1 Broker Kafka

Descrizione dell'architettura e il ruolo del broker Kafka nella pipeline, spiegando come Kafka gestisce i topic, garantendo un flusso di dati affidabile tra le entità della pipeline.



Ogni messaggio inviato deve contenere un identificativo univoco per il paziente, basato sul nome del file di configurazione del paziente e una parte del percorso. Questa modifica garantisce la tracciabilità dei dati e la loro corretta gestione tra i vari componenti del sistema.

Topic	**Descrizione**
breathe-patient-parameters	Dati del paziente, inclusi parametri fisiologici e identificativo univoco (ID)
breathe-ventilator-parameters	Parametri operativi ventilatore.
breathe-feedback	Messaggi di regolazione inviati dal Digital Twin al Physical Twin.
pulse-action	Azioni applicate al paziente durante la simulazione (es. variazione parametri respiratori).

4.2 Formato dei messaggi

Definizione del formato JSON utilizzato per lo scambio dei dati, essenziale per garantire una comunicazione coerente tra i componenti.

Campo	**Descrizione**
timestamp	Data e ora della trasmissione.
source	Identifica il modulo di origine.
type	Specifica il tipo di messaggio (input/output).
topic	Indica il topic su cui è stato inviato il messaggio.
status	Indica se la trasmissione è stata eseguita con successo.
payload	Contenuto del messaggio (JSON).

4.3 Containerizzazione

Tutti i componenti del sistema, inclusi broker, producer e consumer, devono essere eseguiti all'interno di container separati per garantire portabilità, isolamento e scalabilità.

Ogni producer e consumer di dati dispone di un proprio container dedicato. Analogamente, ogni interfaccia che si occupa della comunicazione con un determinato producer o consumer è distribuita in un container autonomo.

Il sistema Kafka è configurato in un cluster composto da quattro container:

- Tre controller gestiscono la ridondanza e la resilienza dei metadati tramite un sistema di elezione a maggioranza.
- Un broker è responsabile della gestione dei topic e dello smistamento dei dati.

La comunicazione tra i container delle interfacce e Kafka avviene tramite porte esposte e configurate secondo il protocollo Kafka (protocollo TCP, porte default 9092 o configurabili).

La comunicazione tra i container delle interfacce e i rispettivi simulatori dipende dal protocollo implementato dal simulatore stesso (es. ZeroMQ o REST), e viene gestita caso per caso.

5. Appendici

5.1 Glossario

Physical Twin: Sistemi fisici o simulatori che generano dati, ovvero il paziente ospedaliero, il ventilatore polmonare.

PT: Abbreviazione di Physical Twin.

Digital Twin: Modelli digitali che replicano i comportamenti dei sistemi fisici.

DT: Abbreviazione di Digital Twin.

ZeroMQ: Middleware per la comunicazione peer-to-peer ad alte prestazioni.

Kafka: Broker di messaggistica distribuito per la gestione di topic.

ETL: Processo di estrazione, trasformazione e caricamento dei dati.

Broker: Sistema intermedio per la trasmissione dei messaggi (Kafka).

RF-'numero': ID relativo ad un requisito funzionale.

5.2 Dataset e messaggi

- Dataset:

  I dataset saranno caricati nel Patient Simulator tramite script Python che convertono i file CSV in messaggi JSON.

  - Respiratory and Heart Rate Dataset v1.0.0: Contiene dati fisiologici di pazienti con patologie respiratorie, utilizzati per addestrare e validare il modello BREATHE.
  - ScienceDirect Respiratory Dataset: Dati reali utilizzati per modellare pazienti con ventilatori, con parametri di frequenza cardiaca e respiratoria.

- Messaggi:

  Di seguito sono inseriti i topic presenti in BREATHE con esempi. Il Server (simulatore) pubblica sulla porta 5555, il client (ventilatore) sulla porta 5556.

  - Server input: questo topic manda due tipi di dati.

    Input iniziale: all'inizio della simulazione viene mandato il contenuto del file di stato utilizzato durante la simulazione (esempi di questi file sono presenti nella cartella BRATHE/breathe.engine/states/). Questi dati contengono tutte le informazioni del paziente allo stato iniziale, sono presenti anche i dati del paziente iniziale nel nodo JSON InitialPatient. Come negli esempi successivi, è presente nel JSON un id per riconoscere il paziente.

Azioni: ogni volta che viene applicata un'azione da parte dell'utente alla simulazione, viene pubblicato il relativo input. Es:


```
{
"id":"states_CSTARS-Patient5@0s.json",
"Action":
      {
      "Airway Obstruction":
            {
            "Severity":0.0
            }
      }
}
```


- Server output: questo topic manda tutti i dati relativi alla simulazione in corso (da quando viene avviata la simulazione fino a quando viene fermata).

  Esempio 1 con solo dati simulazione senza condizioni e azioni applicate:


```
{
"id":"states_CSTARS-Patient5@0s.json",
"Patient Data":
      {
      "SimTime":
            {
            "value": 14.579999999999778,
            "unit": "s"
            },
      "HeartRate":
            {
            "value": 72.01936813828145,
            "unit": "per_min"
            },
      "TotalLungVolume":
            {
            "value": 1652.8755405551394,
            "unit": "mL"
            },
      "RespirationRate":
            {
            "value": 12.048192771084299,
            "unit": "per_min"
            },
      "Lead3ElectricPotential":
            {
            "value": 0.021428571428571443,
            "unit": "mV"
            },
      "CarbonDioxide":
            {
```

```
            "value": 32.24668905630134,
            "unit": "mmHg"
            },
      "ArterialPressure":
            {
            "value": 103.43879910288285,
            "unit": "mmHg"
            },
      "AirwayPressure":
            {
            "value": 760.0,
            "unit": "mmHg"
            },
      "OxygenSaturation":
            {
            "value": 0.9753075079778664,
            "unit": ""
            }
      },
"Conditions": {},
"Actions": {}
}
```


Esempio 2 con dati simulazione dove sono applicate alcune condizioni/azioni.


```
{
"id":"states_CSTARS-Patient5@0s.json",
"Patient Data":
      {
      "SimTime":
            {
            "value": 47.180000000001776,
            "unit": "s"
            },
      "HeartRate":
            {
            "value": 160.34027089291627,
            "unit": "per_min"
            },
      "TotalLungVolume":
            {
            "value": 1709.5003690532112,
            "unit": "mL"
            },
      "RespirationRate":
            {
            "value": 39.47368421052625,
            "unit": "per_min"
            },
```

```
      "Lead3ElectricPotential":
            {
            "value": 0.025315789473684194,
            "unit": "mV"
            },
      "CarbonDioxide":
            {
            "value": 2.7549535343847578,
            "unit": "mmHg"
            },
      "ArterialPressure":
            {
            "value": 78.29280887701968,
            "unit": "mmHg"
            },
      "AirwayPressure":
            {
            "value": 760.0,
            "unit": "mmHg"
            },
      "OxygenSaturation":
            {
            "value": 0.5444834677894493,
            "unit": ""
            }
      },
"Conditions":
      {
      "ARDS":
            {
            "LeftSuperiorLobe Severity": 0.2,
            "RightInferiorLobe Severity": 0.2,
            "RightMiddleLobe Severity": 0.2,
            "LeftInferiorLobe Severity": 0.2,
            "RightSuperiorLobe Severity": 0.2,
            "RightLung Severity": 0.2,
            "LeftLung Severity": 0.2
            },
      "Chronic Anemia":
            {
            "Reduction Factor": 1.0
            },
      "COPD":
            {
            "Bronchitis Severity": 0.3,
            "LeftSuperiorLobe Emphysema Severity": 0.3,
            "RightInferiorLobe Emphysema Severity": 0.3,
            "RightMiddleLobe Emphysema Severity": 0.3,
            "LeftInferiorLobe Emphysema Severity": 0.3,
```

```
            "RightSuperiorLobe Emphysema Severity": 0.3,
            "RightLung Emphysema Severity": 0.3,
            "LeftLung Emphysema Severity": 0.3
            },
      "Chronic Ventricular Systolic Dysfunction": {},
      "Pericardial Effusion":
            {
            "Accumulated Volume": "0.4mL"
            },
      "Pneumonia":
            {
            "LeftSuperiorLobe Severity": 0.6,
            "RightInferiorLobe Severity": 0.6,
            "RightMiddleLobe Severity": 0.6,
            "LeftInferiorLobe Severity": 0.6,
            "RightSuperiorLobe Severity": 0.6,
            "RightLung Severity": 0.6,
            "LeftLung Severity": 0.6
            },
      "Pulmonary Fibrosis":
            {
            "Severity": 0.7
            },
      "Pulmonary Shunt":
            {
            "Severity": 0.8
            },
      "Chronic Renal Stenosis":
            {
            "LeftKidneySeverity": 0.5,
            "RightKidneySeverity": 0.5
            }
      },
"Actions":
      {
      "Airway Obstruction":
            {
            "Severity": 0.5
            }
      }
}
```

- Client input: questo topic manda tutti i dati relativi ai parametri del ventilatore esterno. Questi parametri sono solo quattro in questo caso, essendo il nostro una dimostrazione di un ventilatore a pressione.

```
{
"RR" :
      {
      "unit" : "breaths/min",
      "value" : 12
      },
"PEEP" :
      {
      "unit" : "cmH2O",
      "value" : 5.0
      },
"IE_Ratio" :
      {
      "unit" : "ratio",
      "value" : 0.67
      },
"P_insp" :
      {
      "unit" : "cmH2O",
      "value" : 20.0
      }
}
```

- Client output: questo topic rappresenta I dati che verranno mandati al server (alla simulazione) per la gestione della connessione/ventilazione.

  Richiesta connessione: messaggio per notificare il server di una tentata connessione (mandato solo all'inzio della connessione con il server):

```
{
"message": "requestData"
}
```

  Dati di ventilazione: vengono mandati i dati di ventilazione (o pressione o volume non entrambi):

```
{
"message": "input",
"ventilatorType": "Volume",
"value": "20"
}
```

```
{
"message": "input",
"ventilatorType": "Pressure",
"value": "5"
}
```

Disconnessione: messaggio per notificare il server della disconnessione da parte del server:

```
{
"message": "disconnect"
}
```

5.3 Riferimenti

ISO/IEC/IEEE 29148:2018: *Systems and Software Engineering — Life Cycle Processes — Requirements Engineering*. International Organization for Standardization, Institute of Electrical and Electronics Engineers, 2018. Disponibile su IEEE Xplore: https://ieeexplore.ieee.org/document/8559686/.

Kafka Documentation: *Apache Kafka Documentation*. The Apache Software Foundation, 2024. Disponibile su: https://kafka.apache.org/documentation/.

ZeroMQ Documentation: *ZeroMQ Guide*. The ZeroMQ Project, 2024. Disponibile su: https://zeromq.org/.

Docker Documentation: Best Practices for Containerized Applications. Docker, 2024. Disponibile su: https://docs.docker.com/.

Goldberger AL, Amaral LAN, Glass L, et al.: *Respiratory Dataset v1.0.0*. PhysioNet, 2024. Disponibile su: https://physionet.org/content/respiratory-dataset/1.0.0/.

Bailón R, Sornmo L, Laguna P.: *Respiratory and Heart Rate Dataset*. ScienceDirect, 2024. Disponibile su: https://www.sciencedirect.com/science/article/pii/S2352340923009460/.

Pipeline per Digital Twin. Repository GitHub, 2024. Disponibile su: https://github.com/DaniGreco/Kafka-DigitalTwins/.

Simulatore del corpo umano. Repository GitHub, 2024. Disponibile su: https://github.com/GionathaPirola/BREATHE/.

Foselab: *Kafka-DigitalTwins*. Repository GitHub, 2024. Disponibile su: https://github.com/foselab/Kafka-DigitalTwins/.