MISSION INNOVATION ITALY

MINISTERO DELLA TRANSIZIONE ECOLOGICA

Descrizione dell'architettura del DB

Marco Puccini, Marialuisa Mongelli, Sergio Ferlito, Massimo Celino

ENEA

Report MI21-24/2

DESCRIZIONE DELL'ARCHITETTURA DEL DB

Marco Puccini (ENEA), Marialuisa Mongelli (ENEA), Sergio Ferlito (ENEA), Massimo Celino (ENEA)

Maggio 2022

Report MISSION INNOVATION

Ministero della Transizione Ecologica - ENEA
Mission Innovation 2021-2024 - I annualità
Progetto: Piattaforma accelerata per i Materiali per l'Energia
Work package: IEMAP: Italian Energy Materials Acceleration Platform
Linea di attività: Ambiente di sviluppo HPC: accessi e autenticazione
Responsabile del Progetto: Massimo Celino, ENEA
Responsabile della LA: Marialuisa Mongelli, ENEA

Indice

1 Sommario

Il documento ha come obiettivo riportare lo studio realizzato per pianificare un database per le attività del progetto IEMAP. E' assolutamente necessario realizzare uno studio completo e accurato delle tecnologie disponibili e della gestione dei dati per minimizzare le possibili scelte che possono rivelarsi non ottimali nel corso del progetto. A tal fine si procede con la descrizione delle diverse tipologie di DB e analizzando lo stato dell'arte nel mondo/Italia su come si strutturano i DB per gestire i BigData. Come si vedrà nel seguito, la migliore soluzione di database che soddisfa le necessità di progetto si è trovata nell'infrastruttura MongoDB ENEA di cui sono descritti motivi e vantaggi. Conclude il deliverable una analisi dell'innovatività della soluzione adottata.

2 Introduzione

L'alta eterogeneità dei dati di questo progetto ha richiesto la progettazione di un sistema di archiviazione sul modello architetturale del Data Lake. Sostanzialmente si tratta di un archivio di dati grezzi e non necessariamente strutturati, cui viene aggiunto al livello superiore costituito dai relativi metadati i quali, a livello logico, consentono di costruire l'interoperabilità, l'accessibilità e la ricercabilità degli stessi.

Come emerso nelle attività descritte nel Deliverable D1.7 ("Stato dell'arte nel settore per la Scienza dei Materiali"), requisiti e specifiche dei dati e delle necessità relative ad essi non possono che emergere nel corso del progetto stesso. Questo particolare scenario guida la scelta delle tecnologie da utilizzare per i motivi che descriveremo di seguito, portando la scelta di queste a rivolgersi verso quelle ospitate sull'infrastruttura ENEA.

Tra le molteplici tecnologie per database una prima, seppur superficiale distinzione si può fare tra quelli relazionali e quelli non relazionali. Mentre tra i primi raccolgono fondamentalmente diverse implementazioni di archivi basati sullo schema entità-relazioni interrogabili per mezzo di un determinato linguaggio, ovvero database SQL (Structured Query Language), nel sottoinsieme dei database non relazionali vengono raggruppate tecnologie anche molto distanti tra loro. Sono non relazionali, ad esempio, tanto i database a grafo quanto quelli documentali, pur seguendo logiche completamente diverse in merito alla rappresentazione del dato. Inoltre, a parità di tipologia di database, esistono differenti tecnologie che la implementano; dunque, esistono per esempio diversi database documentali con caratteristiche e specifiche molto differenti. Mentre i database relazionali hanno dominato la scena per anni, dall'avvento di alcune tecnologie come il web 2.0, i social network, l'e-commerce, l'IoT, i Big Data ed il mondo dell'AI si sono diffuse e moltiplicate le tecnologie non relazionali, nate proprio per rispondere alle nuove sfide.

Non vuole essere oggetto di questo documento una disamina dettagliata di questo ampio e complesso mondo, ma occorre sottolineare come vi siano delle soluzioni che meglio di altre si prestano ad adattarsi a scenari differenti e anche, soprattutto, mutevoli nel tempo. Questo è il caso del DBMS mongoDB, un database documentale, il più diffuso tra i non relazionali ed ampiamente adottato in molti degli ambiti sopra citati. Proprio per questi motivi è stata la scelta tecnologica adottata da ENEA per gestire diversi progetti di ricerca, offrendo il supporto di un cluster dedicato per gestire scenari molto diversi di acquisizione e gestione dei dati.

Un elemento strategicamente rilevante della scelta di questa soluzione tecnologica risiede nella sua estrema flessibilità e adattabilità anche in corso d'opera, *feature* per la quale questa soluzione è la preferita in ambito sviluppo, quando non necessariamente esistono requisiti e specifiche fin da subito. Soprattutto con i moderni approcci e metodologie di sviluppo software come il cosiddetto metodo *Agile* e le procedure *DevOps*, una tecnologia simile si presta ad offrire una soluzione affidabile in ogni step, consentendo di modificare in corso lo schema dei dati senza mai per questo interrompere il servizio.

3 Descrizione delle attività svolte e risultati

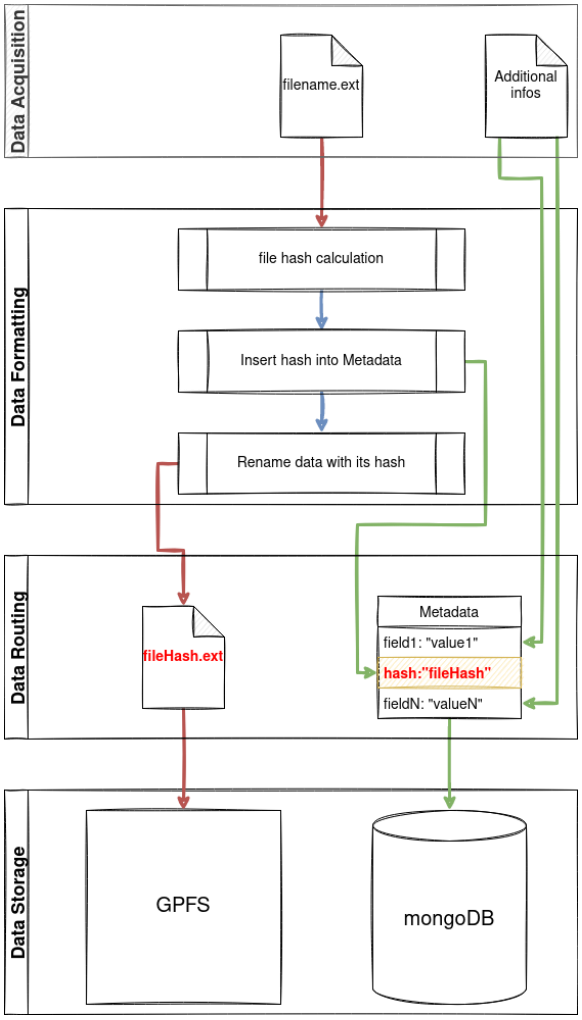
3.1 Data Lake per IEMAP

Il Data Lake implementato da ENEA per IEMAP, costituisce il primo e fondamentale layer software della piattaforma. Qualunque servizio e/o attività di analisi, sviluppo, test riguarda infatti i dati come elemento primario, generatore di conoscenza, sorgente di qualsiasi informazione che voglia e debba essere ricercata, correlata, sperimentata o verificata.

Lo schema organizzativo del flusso di informazioni, dalla loro acquisizione alla loro archiviazione, consta di quattro passaggi, per ogni esperimento/simulazione:

1. **Data Acquisition**: vengono raccolti:
    a. i dati, ossia i file in cui sono raccolti i risultati sperimentali e/o i file di output generati nella simulazione;
    b. Un set di informazioni aggiuntive sull'esperimento/simulazione, su chi li ha condotti, nell'ambito di quale progetto, per quale istituzione, etc.;
2. **Data Formatting**: le informazioni raccolte al punto 1.b vengono formattati secondo lo schema descritto nel Deliverable D1.7 e serializzati nel formato JSON, per ottenere un documento contenente i metadati. Inoltre, viene:
    a. calcolato l'hash dei file al punto 1.a, ottenendo una stringa alfanumerica unica;
    b. inserita questa stringa all'interno del documento JSON dei metadati;
    c. Rinominato il file del punto 1.a con il suo hash (mantenendo l'estensione originale);
3. **Data Routing**: a questo punto tutte le informazioni (dati e relativi metadati) vengono instradati verso i sistemi di storage definiti al punto successivo;
4. **Data Storage**: le informazioni vengono archiviate come segue:
    a. I dati (i file) in un'area dedicata del File System distribuito GPFS di ENEA;
    b. I metadati (JSON) in un database dedicato all'interno dell'infrastruttura mongoDB di ENEA

Di seguito viene illustrato lo schema logico del Data Lake descritto:



Le operazioni descritte nel blocco *Data Formatting* sono quelle che consentono:

1. L'identificazione univoca dei dati all'interno del Data Lake;
2. La trasportabilità delle informazioni (i file potrebbero essere spostati su altri sistemi di storage, ad esempio un Object Storage, rimanendo identificati all'interno del database);
3. L'interoperabilità, grazie allo schema con cui vengono organizzati i metadati, coerente per ogni tipologia di dato

Da sottolineare un aspetto non scontato della logica descritta: è stata pensata in funzione di una sua automazione. Non si tratta dunque di automatizzare una prassi "manuale", quanto strutturare tale prassi affinché sia ottimizzata la sua automazione. Questo comporta che alcuni passaggi possano apparire delle complicazioni, ma solo se si pensano effettuati da un operatore umano. L'intero workflow, infatti, è pensato per essere eseguito da delle API (Application Programming Interface) il cui sviluppo è stato già avviato. Tali API si prenderanno in carico di svolgere le operazioni descritte, gestendo anche tutta la logica delle autenticazioni, delle autorizzazioni, delle concorrenze negli accessi, della trasmissione delle informazioni, delle sessioni attive e la loro persistenza, della sicurezza dei protocolli di trasmissione impiegati.

Grazie alle API sarà possibile lo sviluppo dell'interfaccia del portale secondo gli standard di sicurezza informatica vigenti e le best practice in termini di architettura software di web application. Oltre all'interfaccia della piattaforma, le API offrono chiaramente uno strumento di interazione utile a sviluppatori e ricercatori che volessero automatizzare ulteriormente l'interazione con il Data Lake (richiesta di dati per training e testing di modelli di Deep Learning, correlazioni, invio di flusso di dati).

L'interfaccia della piattaforma consentirà dunque di caricare i file di esperimenti/simulazioni, raccogliendo i metadati attraverso un *form* che guiderà il ricercatore nella sua compilazione. Questa interfaccia passerà le informazioni alle API che avvieranno il workflow descritto, archiviando tutte le informazioni.

3.2 Sistemi di archiviazione

Ogni file caricato sulla piattaforma IEMAP, viene gestito dalle API come descritto nel paragrafo precedente, venendo archiviato sul file system distribuito IBM Spectrum, in un'area dedicata ed ospitata sull'infrastruttura GPFS di ENEA, sita nel Centro di Ricerche Portici di ENEA.

GPFS è un file system POSIX con gestione dei permessi Unix-like, ospitato da un cluster di N macchine ed accessibile mediante diversi protocolli di comunicazione.

Ogni documento JSON contenente i metadati relativi ad ogni dato, viene archiviato sull'infrastruttura mongoDB di ENEA. Tale infrastruttura, per dimensioni e caratteristiche architetturali, è rivolto alle attività di ricerca offrendo una soluzione moderna per l'archiviazione e la gestione dei dati. Nonostante le dimensioni ridotte in termini di numero di macchine, è stato configurato ed installato per garantire alle suddette attività i benefit tipici di questa tecnologia, in particolare:

- *Partition tolerance*: il sistema garantisce il funzionamento anche in caso uno dei nodi dovesse fermarsi o interrompere le sue comunicazioni con gli altri;
- *Replication*: ogni dato viene replicato su un numero dispari di nodi, minimo 3, che costituiscono un pool di macchine definito *Replica Set*;
- *Schema on-read*, *schema versioning* ed in generale differenti *schema pattern*: la struttura con cui si organizzano i dati archiviabili nel database è estremamente flessibile. Questa può essere definita addirittura al momento in cui vengono richiesti (particolarmente utile nel caso dell'analisi dati), consentendo di essere raccolti così come sono prodotti (dai social, da strumenti di misura, da sensoristica, …). Tale struttura può essere modificata nel tempo, riorganizzando quella esistente o stravolgendola senza compromettere il comportamento del sistema (particolarmente utile nelle fasi di sviluppo, quando non necessariamente sono noti i requisiti con cui devono essere poi utilizzati i dati);

A questi ed altri aspetti tecnici si aggiungono i vantaggi di essere una tecnologia molto diffusa, con un'ampia community di utenti e di sviluppatori. Essendo infine un software sostanzialmente open source, almeno nelle sue componenti principali, risulta essere particolarmente vantaggioso anche in termini economici e di sicurezza informatica, essendo il codice costantemente messo alla prova dall'intera comunità di computer scientists.

Entrando nel dettaglio tecnico del cluster di ENEA, è necessario introdurre alcuni elementi sui quali si basa l'architettura del sistema. Come precedentemente accennato, il Replica Set è l'insieme dispari di macchine che garantisce la replica del dato e rappresenta l'entità principale con la quale si costruisce un'architettura

mongoDB. Il numero di macchine di un Replica Set è fissato dagli algoritmi che gestiscono il consenso che, per evitare problemi di Brain split, deve essere dispari. All'interno di un Replica Set, infatti, le macchine possono avere sostanzialmente due funzioni:

Primary: è la macchina che riceve le richieste di lettura e scrittura del dato

Secondary: sono le altre macchine, sulle quali viene replicato il dato una volta scritto

Quando il Primary fallisce, i secondary devono eleggerne uno nuovo tra loro. Questo anche nel caso in cui il Primary dovesse semplicemente smettere di comunicare con il resto del Replica Set: in questo caso, infatti, il Primary si auto-declassa a Secondary, lasciando la maggioranza del pool decidere un suo sostituto, fintanto che non vengono ripristinate correttamente le connessioni. Per la natura stessa degli algoritmi di gestione del consenso in questi sistemi - in particolare mongoDB ne utilizza uno denominato *Raft* - il numero di macchine deve essere 2n+1.

Il motivo per cui il cluster di ENEA riesce ad assicurare le caratteristiche precedentemente elencate risiede nella sua architettura che implementa, seppure in una configurazione minimale, il concetto di Sharding. Lo Shard è un partizionamento orizzontale dei dati, che vengono scomposti (come un puzzle) e distribuiti su più macchine, distribuendo su queste il volume dei dati. Per consentire questo comportamento sono necessari due componenti, costituiti ciascuno da un Replica Set:

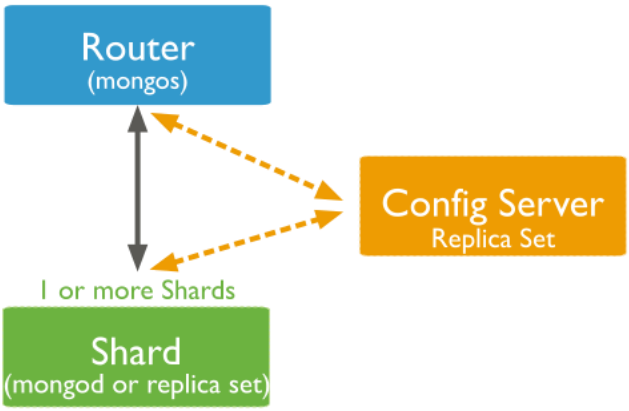
Lo Shard: un Replica Set che gestisce i dati. Aggiungendo nuovi Shard si ottiene la scalabilità orizzontale del sistema, sia in termini di storage che di prestazioni e accessibilità;

Il Config Server: un altro Replica Set che gestisce i log di tutte le operazioni, tenendo traccia di tutte le richieste di lettura e scrittura, della mappa dei singoli pezzi di dato e della loro collocazione nei diversi Shard e dunque dell'istradamento delle richieste per ricostruire il dato originale, andando a recuperarne i vari pezzi (hash).

Ovviamente un cluster che voglia sfruttare lo Sharding del dato deve avere più Shard, tuttavia quello di ENEA ne possiede solo uno. Anche se apparentemente può risultare un controsenso, in realtà l'aver configurato l'architettura anche con un singolo Shard ne consente l'ampliamento e dunque renderlo potenzialmente scalabile. Cosa che non potrebbe essere possibile qualora non fosse stato configurato fin dall'inizio in questo senso. È ovviamente possibile convertire un cluster non Sharded a *Sharded Cluster*, ma questo comporta necessariamente operazioni complesse e l'interruzione del servizio. In questa configurazione invece, il sistema è pronto a scalare senza interruzioni.

Come è stato detto, l'altro componente che consente ad un cluster di fare Sharding è il Config Server. Poiché il cluster ENEA è composto da sole tre macchine, è stato adottato un espediente che consente di ospitare sulle stesse macchine sia il Replica Set di uno Shard, sia quello del Config Server. In particolare, su ogni macchina del cluster, sono operativi due servizi *mongod*, gestiti dal systemd di sistema, configurati l'uno per servire lo Shard, l'altro il Config Server. Essendo dunque 2 Replica Set sulle stesse 3 macchine, onde evitare che con il fallimento di una si determini la *caduta* di entrambi i Primary (potenzialmente), questi sono stati sfasati.

A questo pool di 3 macchine se ne aggiungono ulteriori 2 che ospitano semplicemente il servizio mongos, ovvero il routing di mongoDB. Questo si incarica dell'interfacciamento tra utente e cluster, dialogando con il Config Server, gli comunica le richieste dell'utente e restituisce i risultati. Sostanzialmente funge da frontend al sistema. Di seguito è raffigurato lo schema del cluster mongoDB di ENEA:



Il cluster infine è configurato con il sistema di autenticazione interno di mongoDB, consentendo le connessioni e le interazioni con questo, soltanto alle istanze autenticate.

A livello di organizzazione logica dei dati, internamente al DBMS, mongoDB suddivide i dati in Database, i quali possono contenere una o più collezioni, le quali raccolgono di documenti in formato JSON, che altro non sono che i dati. Il cluster di ENEA ospita dunque diversi database dedicati a diversi progetti, ognuno dei quali contiene un numero di collezioni definite autonomamente dal progetto, con relative utenze. In particolare, si possono definire un numero indefinito (il limite superiore è rappresentato dallo storage a disposizione) di utenze, ciascuna delle quali può avere determinati permessi a seconda delle operazioni che si vogliono consentire. Tali permessi possono essere definiti per mezzo di raggruppamenti predefiniti (*Built-in Roles*) o customizzati all'occorrenza (*User-defined Roles*).

Per il progetto IEMAP è stato dunque predisposto un Database dedicato sull'infrastruttura mongoDB di ENEA, con utenti specifici e permessi specifici a seconda del tipo di operazioni da fare con i dati.

Il database è al momento accessibile esclusivamente da rete ENEA. Tuttavia, l'accesso sarà gestito a livello di portale, dove sarà prevista un'apposita sezione di registrazione che consentirà di accedere sia alle pagine di consultazione del Data Lake che al form di caricamento dati. Se dunque è possibile accedere per verifiche sul database, per motivi di sicurezza informatica, la prassi che sarà ritenuta ufficiale e dunque promossa e descritta in successivi Deliverable, sarà quella citata mediante portale.

Per chi volesse, a scopo valutativo e/o esplorativo, verificare le caratteristiche del database e dell'infrastruttura descritti, potrà inoltrare richiesta al responsabile amministratore dell'infrastruttura e DBA (Data Base Administrator), all'indirizzo mail marco.puccini@enea.it che rilascerà un account di sola lettura ed il dettaglio sulle modalità di accesso, che sono indicate principalmente attraverso due strumenti:

1. Il client grafico ufficiale di mongoDB, scaricabile all'indirizzo:

   https://www.mongodb.com/try/download/compass

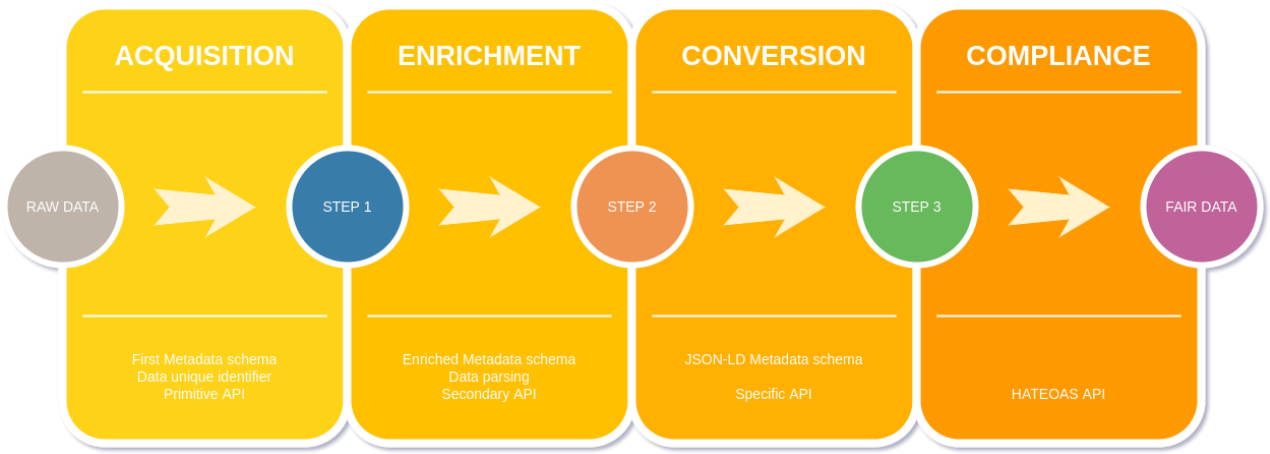
2. Il tool da riga di comando (CLI: Command Line Interface) ufficiale, scaricabile all'indirizzo:

   https://www.mongodb.com/try/download/shell

Una volta implementate le API ed il *frontend* del portale, si avrà una completa fruizione dei dati e servizi del Data Lake. È doveroso specificare inoltre che gli strumenti appena indicati consentono la connessione esclusivamente al database e dunque ai metadati ivi contenuti. L'accesso ai dati grezzi ospitati su GPFS è invece inibito, per scoraggiare gli utenti dal modificare e quindi, potenzialmente, rompere la consistenza tra dati e metadati. Anche in questo caso, attraverso il portale e quindi le API, sarà possibile scaricare i file di interesse per analisi e processazioni.

4 FAIRness dell'architettura di gestione dei dati

L'architettura hardware, software, lo schema dei metadati presentati fin qui, sono frutto di un *trade-off* tra l'esigenza di iniziare ad acquisire nel minor tempo possibile i dati, elemento strategico, con la necessità di organizzarli e presentarli seguendo i principi FAIR [1]. In tal senso, molte delle scelte effettuate sono parziali implementazioni tecniche e logiche di tali principi, nell'ottica di una completa *compliance* secondo un percorso incrementale. Tale roadmap è schematizzata di seguito:



Attualmente è stato raggiunto lo STEP 1, con:

1. Metadati: un primo schema relativo ai metadati,
2. Dati: la creazione di un identificatore univoco per i dati inserito nei metadati come descritto precedentemente,
3. API: la progettazione ed il parziale sviluppo delle API con l'implementazione di funzioni primitive di accounting, autorizzazione, inserimento dei metadati, upload dei dati, prime ricerche sul database.

Gli step successivi prevedono avanzamenti sui tre assi principali per un graduale raggiungimento degli standard richiesti. In particolare:

- STEP 2
    - Metadati: arricchimento delle informazioni inserite tra i metadati, a partire da quelle contenute nei dati
    - Dati: parsing dei dati per l'estrazione delle informazioni di cui al punto precedente

  - API: Implementazione di funzioni di ricerca più avanzate nelle API

- STEP 3
  - Metadati: Conversione dei metadati implementando la sintassi JSON-LD [2] per linked data
  - API: Implementazione di ulteriori funzioni avanzate nelle API
- FAIR DATA
  - API: L'adeguamento allo standard HATEOAS (Hypermedia as the Engine of Application State) per le API [3], in particolare l'inserimento della lista dinamica degli altri endpoint consultabili nel risultato di ogni singolo endpoint, in funzione del dato richiesto.

Riguardo l'attuale stato di avanzamento, si possono sottolineare i risultati raggiunti in termini dei quattro principi FAIR, con relativi sotto-principi, riportati per esteso.

4.1 Findable

- **F1. (Meta)data are assigned a globally unique and persistent identifier**: come descritto nella sezione 2, i dati vengono rinominati con la stringa di hash calcolata sul loro contenuto. Questo rende la stringa univocamente legata al dato. I metadati sono univocamente identificati dall'ID creato dal database al momento del loro inserimento.
- **F2. Data are described with rich metadata (defined by R1 below)**: lo schema con cui sono organizzati i metadati, mira esattamente a questo scopo.
- **F3. Metadata clearly and explicitly include the identifier of the data they describe**: la stringa di identificazione univoca del dato, viene inserita all'interno dei metadati come valore in chiaro.
- **F4. (Meta)data are registered or indexed in a searchable resource**: i riferimenti ai dati sono inseriti all'interno dei metadati come campi indicizzati nel database, mentre i metadati sono archiviati ed indicizzati sempre nel database.

4.2 Accessible

- **A1. (Meta)data are retrievable by their identifier using a standardised communications protocol**: attraverso l'utilizzo delle REST API, i dati e metadati sono resi accessibili attraverso il protocollo standard HTTP utilizzando dunque i relativi metodi.
  - **A1.1 The protocol is open, free, and universally implementable**: il protocollo HTTP rispetta i criteri citati.
  - **A1.2 The protocol allows for an authentication and authorisation procedure, where necessary**: le API prevedono l'implementazione anche di procedure di autenticazione e autorizzazione mediante lo standard JWT (JSON Web Token) e Bearer.
- **A2. Metadata are accessible, even when the data are no longer available**: l'architettura di mongoDB garantisce l'accessibilità ai metadati anche qualora i dati fossero non raggiungibili.

4.3 Interoperable

- **I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation**: il formato di serializzazione dei metadati è lo standard JSON. In previsione sarà adottata la sintassi JSON-LD come esplicitamente richiesto nelle specifiche per le API dei FDP (FAIR Data Pont) [4].
- **I2. (Meta)data use vocabularies that follow FAIR principles**: Nell'adozione della sintassi JSON-LD, si prevede di implementare ontologie già utilizzate nella scienza dei materiali, eventualmente implementandone di dedicate, mappando i riferimenti a quelle già note.
- **I3. (Meta)data include qualified references to other (meta)data**: anche questo punto sarà rafforzato dall'adozione della sintassi JSON-LD e delle ontologie dedicate.

4.4 Reusable

- **R1. (Meta)data are richly described with a plurality of accurate and relevant attributes**: il dettaglio delle informazioni raccolte nei metadati è descritto nel Deliverable D1.7, precedentemente citato. Queste, sommariamente, descrivono in modo approfondito il dato cui si riferiscono, sotto molteplici aspetti. Dalla proprietà, al contesto in cui è stato prodotto, al suo contenuto scientifico.
  - **R1.1. (Meta)data are released with a clear and accessible data usage license**: le API di accesso ai dati e metadati, sono corredate da un'accurata documentazione tecnica (in corso di redazione).
  - **R1.2. (Meta)data are associated with detailed provenance**: la provenienza del dato viene dettagliata in un subset di informazioni incluse nei metadati.
  - **R1.3. (Meta)data meet domain-relevant community standards**: attualmente lo schema dei metadati è stato redatto facendo riferimento alle specifiche emerse dalla survey condotto con i partner di progetto, esperti di settore. La particolare necessità di far confluire dati sperimentali e computazionali ha richiesto come passaggio intermedio quello attuale, per formalizzare i campi specifici ma in uno schema omogeneo. Successivamente, l'adozione di ontologie specifiche per i due domini, dovrà probabilmente prevedere la definizione di vocabolari in grado di astrarre un livello superiore di inclusione.

5 Conclusioni

Le attività di acquisizione, archiviazione e gestione dei dati hanno avuto la massima attenzione nel progetto, essendo la base di partenza della piattaforma che si vuole sviluppare. Tuttavia, quanto descritto, rappresenta solo il primo strato di servizi di backend necessari. Senza un sistema che consenta l'interazione agevole con il cluster mongoDB e quello GPFS, risulta difficile poter pensare che un utente o un sistema automatico possa interagirvi, inviando o richiedendo dati.

Per questo motivo, parallelamente alle attività descritte, sono state avviate anche quelle di progettazione e sviluppo di API RESTful e GraphQL che consentiranno di essere interrogate dal frontend della piattaforma, offrendo le funzionalità di caricamento ed interrogazione dei dati.

La scelta di procedere a sviluppare ogni singolo componente separatamente, in modo simile a quanto accade nelle architetture a microservizi, consente procedere anche parallelamente ed in modo indipendente, mantenendo un coordinamento unico per non perdere ovviamente la coerenza tra i sistemi. Il vantaggio è la maggiore capacità di modellare le soluzioni ai problemi e requisiti che emergono avanzando con il progetto, ad una più efficace gestione delle manutenzioni e degli aggiornamenti, alla possibilità di estendere la piattaforma con servizi aggiuntivi senza dover ridisegnare ogni volta l'intero sistema.

6 Riferimenti bibliografici

[1] Mark D. Wilkinson; et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data, 15 March 2016. DOI: doi:10.1038/sdata.2016.18

[2] Marco Pritoni, Drew Paine, Gabriel Fierro, Cory Mosiman, Michael Poplawski, Avijit Saha, Joel Bender and Jessica Granderson, *Energies* 2021, *14*(7), 2024; https://doi.org/10.3390/en14072024

[3] https://specs.fairdatapoint.org/

[4] https://www.fairdatapoint.org/

7 Abbreviazioni ed acronimi

DB = DataBase

DBA = Data Base Administrator

FAIR (dati) = dati che rispettano i principi di rintracciabilità (findability), accessibilità (accessibility), interoperabilità (interoperability) e riusabilità (reusability)

API = Application Programming Interface

FRD = Fair Data Point

JSON = JavaScript Object Notation

CLI = Command Line Interface

GPFS = General Parallel File System

POSIX = Portable Operating System Interface for Unix

SQL = Structured Query Language