ISTITUTO NAZIONALE DI FISICA NUCLEARE

Direzione Sistemi Informativi

INFN-23-11-DSI
5 aprile 2023

Sistema Gestionale dei Prodotti della Ricerca: CDR e WBS di Progetto

Antonello Paoletti[1], Marzio D'Alessandro[1], Mauro Gattari[1],
Francesco Serafini[1], Luca Sanelli[1]

[1)] *INFN, Direzione Sistemi Informativi, I-00044 Frascati (Roma), Italy*

Abstract

Sviluppo di un sistema **gestionale** per l'archiviazione di metadati e documenti relativi a prodotti della ricerca INFN. Il sistema qui descritto è parte di un progetto più ampio, coordinato con il gruppo OpenScience e il Gruppo di Lavoro sulla Valutazione dell'INFN mirato allo sviluppo di un CRIS (Current Research Information System) d'Istituto.
Il sistema **gestionale** è progettato per fornire servizi di archiviazione e gestione dei dati relativi a prodotti della ricerca, interoperando con servizi e banche dati interni (GODiVA, Zenodo, GLVsoft) ed esterni (Clarivate, Scopus, DOI) al fine di costruire un *repository* autoritativo e integrato a supporto di attività gestionali, di archivio, divulgazione, monitoraggio e valutazione.
Questo lavoro nasce con il superamento del concetto di "Database delle Pubblicazioni", inteso come *mirror* o archivio di metadati e punta, invece, sullo sviluppo di un sistema di "transito" e arricchimento delle informazioni nell'ambito di una rete di "produttori" e "consumatori" di dati sulla produttività scientifica dell'Ente.
Tale sistema ha la capacità di acquisire, far coesistere e collegare informazioni provenienti da fonti diverse, al fine di generare un patrimonio informativo complesso attraverso interfacce (UI/API) e meccanismi semi-supervisionati di *machine learning*.

DOI n. 10.15161/oar.it/76981

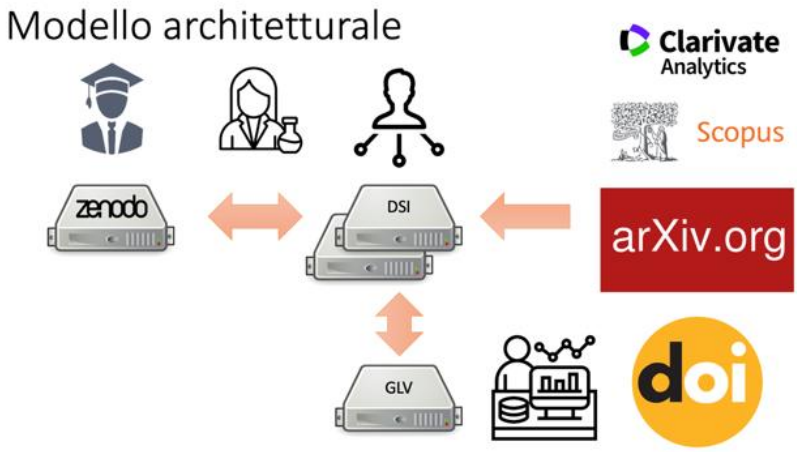
Published by
Laboratori Nazionali di Frascati

1 Motivazioni

La mancanza di un CRIS[1] "ufficiale" ed unico ha portato l'INFN a dotarsi di molteplici *repository* di prodotti della ricerca non integrati, con esigenze divergenti e soluzioni poco sostenibili nel tempo. Le **VQR**[2] hanno acuito questa criticità, facendo emergere la necessità di un sistema altamente *flessibile* e integrato, in grado di processare informazioni sui prodotti della ricerca con modalità molto variabili e difficilmente anticipabili o codificabili.

Si è deciso, quindi, di riprogettare l'archivio dei prodotti della ricerca ponendo l'accento sulla **flessibilità** nella modellazione del dato gestito e sulla qualità di **servizi** e **interfacce** per acquisire, consolidare e **arricchire** le informazioni sui prodotti. Una sfida, fra le tante, consiste nel far convivere in maniera funzionale ed efficiente le informazioni provenienti da fonti eterogenee (es. *Scopus*[3]e *arXiv*[4]) all'interno di strutture dati flessibili e **collegate**, con una gestione del ciclo di vita orientata a massimizzare la trasparenza, la coerenza e la fruibilità del dato.

Per ottenere questo, occorre definire processi gestionali a supporto del ciclo di vita di un prodotto della ricerca, progettando servizi, interfacce utente ed *Application Programming Interface (API)*[5]che favoriscano l'interazione con utenti finali e sistemi terzi, al fine di gestire il dato ma anche esplicitarne le connessioni interne (es. preprint vs VOR[6]) ed esterne (es. riconoscimento autori/progetti) al *dataset* originale.



L'obiettivo è di costituire un patrimonio informativo complesso e organico che "racconti" la produzione scientifica sia attraverso i dati ma anche e soprattutto attraverso le

[1] https://en.wikipedia.org/wiki/Current_research_information_system
[2] https://www.anvur.it/attivita/vqr/
[3] https://www.scopus.com/
[4] https://arxiv.org/
[5] https://it.wikipedia.org/wiki/Application_programming_interface
[6] https://royalsociety.org/journals/ethics-policies/media-embargo/

interrelazioni fra di essi. Una delle sfide illustrate in questo lavoro presenta le opportunità fornite da modelli di *machine learning* per far emergere tali relazioni con modalità non supervisionate o semi supervisionate.

Tale patrimonio sarà messo infine a disposizione di servizi terzi per ulteriori elaborazioni, continuando nel ciclo di vita (e di "arricchimento") del dato, acquisendone i risultati in forma di metriche o nuovi metadati.

2 Organizzazione e WBS

È di seguito illustrata la WBS di progetto organizzata per *Work Package* (con impatto sullo sviluppo) composti da attività ad alta coesione interna. Le attività sono tarate per avere un basso *coupling* e massimizzare l'avanzamento in parallelo degli sviluppi:

- *WP0: Organizzazione e coordinamento con OS, GLV, DSR, CSN* (A. Paoletti, F. Serafini)
  - Sfide: mettere a sistema le competenze tecniche e di dominio fra le diverse realtà dell'Ente al fine di massimizzare l'efficacia e la semplicità d'uso del prodotto
- WP1: Definizione del *datamodel* e del ciclo di vita del dato (M. D'Alessandro)
  - Sfide: definire un modello coerente ed esaustivo dei metadati di un articolo che permetta di far convivere informazioni con codifiche disomogenee
- WP2: Sviluppo dei connettori verso fonti di metadati (M. D'Alessandro)
  - Sfide: sviluppare sistemi altamente resilienti che permettano di acquisire dati da fonti esterne all'INFN in relazione al *datamodel* definito
- WP3: Sviluppo del motore di *machine learning* (M. Gattari)
  - Sfide: fornire un servizio semi-supervisionato che permetta di elaborare informazioni e stabilire relazioni fra i dati in archivio
- WP4: Sviluppo di API (L. Sanelli)
  - Sfide: definire modelli di interazione con servizi "clienti" del sistema gestionale per attività di *enrichment* delle informazioni in archivio
- WP5: Sviluppo di un servizio di riconoscimento e aggancio autori (L. Sanelli)
  - Sfide: definire modelli di gestione delle stringhe autori per collegarli a identità in archivio
- WP6: Sviluppo dell'interfaccia utente (M. Gattari)
  - Sfide: definire maschere e processi di interazione con l'utente finale in linea con le *best practices* in tema di interfacce utente ed esperienza utente

3 Processo e attori coinvolti

La platea potenziale del sistema gestionale coinvolge tutto il personale di ricerca, prescindendo sia dal livello contrattuale che dal ruolo scientifico o di organigramma, e includendo tutti coloro che possono dare un contributo all'archivio istituzionale. Dal tesista, al ricercatore, fino al responsabile d'esperimento e al membro di Giunta, l'informazione nasce e si modifica secondo livelli di accesso via via crescenti che si

esauriscono con il management, il gruppo di valutazione e le Commissioni Scientifiche Nazionali (CSN).

Il processo di gestione di un prodotto ne riflette il ciclo di vita, a partire da un "manoscritto", edito da uno o più autori (AAM o *preprint)*, per poi concludersi con la pubblicazione su rivista o su archivi istituzionali. Tale processo viene "specializzato" in base al tipo di prodotto, definendo passaggi, interfacce e *dataset* a seconda della natura del dato e dell'attore che lo sta gestendo.

La "specializzazione" dei processi ha impatto sin dal *data entry*, individuando in strumenti interni (Zenodo, UI ad hoc) e/o esterni (API *Clarivate*/*Scopus*) la "porta d'ingresso" per le diverse tipologie di prodotto. Sono proprie di questa fase le criticità relative all'**accuratezza** del dato, sia in relazione alla numerosità (*recall*) che alla *purezza*[7] e all'attendibilità (vedi *bibliometria*[8]). Per gestire queste problematiche, i processi di acquisizione sono codificati individuando "fonti" autorevoli (ad es. *Scopus*) su cui sono definiti sottoprocessi gestionali che governano l'acquisizione, la trasformazione, la conservazione e la fruizione del dato gestito. Tali sottoprocessi sono supportati sia da servizi automatici di *data loading* che da UI per la *data curation*[9] ad opera di personale bibliotecario specializzato. Tutti i processi di acquisizione si concludono con il popolamento del "modello di prodotto", una struttura dati generalizzata e flessibile, con un set minimo di metadati e che operi come "indice" verso le informazioni originali acquisite dalle fonti.

La fase successiva, più puramente *gestionale*, riguarda l'arricchimento del dato ed è la più complessa. Anche questa fase coinvolge attori "umani" e servizi (pseudo) automatizzati, che operando di concerto sul modello, dovranno cooperare garantendo che l'informazione non perda la sua coerenza e il significato originario. A tal proposito, sono definite strutture di *logging* e *versioning* sul modello, al fine di identificare ed eventualmente annullare le operazioni fatte da un utente o da un servizio. Rientra in questa fase la gestione del *dataset* del modello (metadati, allegati e metriche) con la possibilità di estenderlo, modificarlo e valorizzarlo sia da UI che tramite API ad es. definendo un voto ai fini della VQR. Un obiettivo importante è produrre valore sul dato acquisito, contestualizzandolo rispetto ad altri servizi e banche dati ed esplicitando relazioni verso questi ultimi ad es. collegando il prodotto ad un altro prodotto o collegandolo a un progetto INFN o ad un'identità digitale.

[7] https://towardsdatascience.com/accuracy-precision-recall-5c8b8f0abde1
[8] https://it.wikipedia.org/wiki/Bibliometria
[9] https://it.wikipedia.org/wiki/Data_curation

4 Modello concettuale

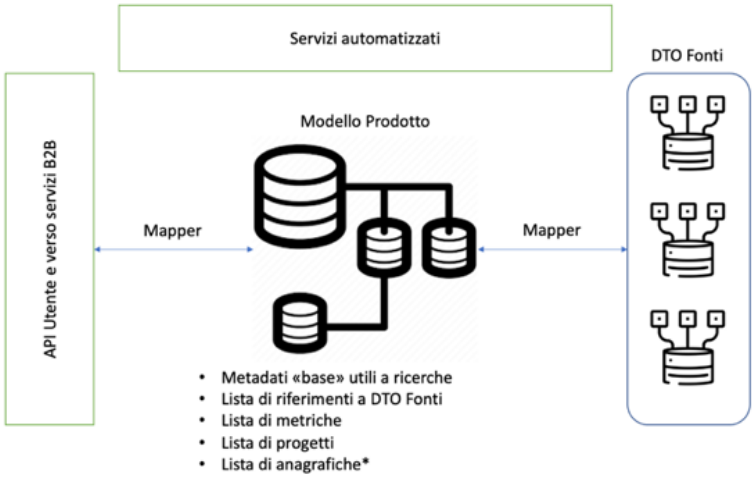
La sfida principale riguarda il ciclo di vita e il modello dati delle entità gestite. Trattandosi di un sistema di "transito" e non di mera "conservazione", occorre modellare il dato introducendo alti livelli di flessibilità nel modo in cui viene acquisito, trasformato e reso fruibile.

L'entità principe di questo lavoro è il "prodotto della ricerca", inteso come insieme di **metadati** e allegati relativi a un'attività di ricerca documentata. Esempi di prodotto sono gli articoli scientifici su rivista, i contributi a conferenza e le tesi. Ognuno di questi elementi è caratterizzato da informazioni testuali e metriche che ne definiscono il valore informativo intrinseco e relazioni verso altri elementi o banche dati (es. sigle di progetto) che lo "collocano" in un più ampio patrimonio informativo.

La struttura logica di un "prodotto" è comune in molte sue parti e peculiare allo stesso tempo negli elementi tipici di ogni tipologia. Ad es.: in un articolo scientifico non ha importanza il campo "Università", a differenza di una tesi, mentre il campo "Autore" ricorrerà in entrambi i tipi. È quindi fondamentale individuare un modello che si adatti al dato gestito e possa mantenere quanto più possibile inalterato il significato originario, arricchendolo all'occorrenza.

Su tale modello è definito il processo di arricchimento e gestione del dato, a seguito di "esposizione" dello stesso all'utenza, a servizi automatizzati o di terze parti. La criticità è principalmente derivata da un non completo controllo sui (sotto) processi gestionali extra sistema. In questo caso, sarà d'obbligo stabilire e concordare interfacce/politiche di condivisione del dato al fine di mantenere una coerenza interna e filtrare quanto più possibile l'entropia generata dall'utente. Ad es. è buona norma stabilire regole di integrità ("almeno un autore definito") o definire *Data Transfer Object (DTO)* limitati sulle operazioni permesse via *Application Programming Interface (API)*.

L'ultimo elemento riguarda la fruibilità del dato sia tramite interfacce di UI che tramite API o sistemi di *datawarehouse* e *business intelligence.* Su questo fronte è fondamentale identificare e perimetrare le necessità di chi userà il dato al fine di strutturare elementi intermedi (*DTO*) in grado di rimodulare il dato secondo le necessità del momento. Il concetto chiave è di non limitare la progettazione del modello secondo un'esigenza specifica ma prepararlo a gestirne di molteplici, trasformandosi di volta in volta. Ad es. il *DTO* di un prodotto fornito ad una UI sarà diverso dal *DTO* esposto alle API verso servizi terzi o a servizi di *ETL*.



5 Tecnologie

Il *core* del sistema è costituito dal **sistema gestionale**, progettato per operare come aggregatore di informazioni fra fonti dati diverse. Il gestionale ha il compito di interagire sia con servizi terzi (interni ed esterni) sia con l'utenza finale, articolando il suo funzionamento in attività di *backend* e *frontend* con una netta separazione fra i due ambiti. Tale separazione si articola sia sul piano concettuale (tipologia di attori, casi d'uso…) sia sul piano tecnologico.

Il sistema di *backend* è strutturato come un (micro)servizio REST sviluppato in *JAVA* sul framework *SpringBoot* e sulla libreria **BaseLib** del sistema informativo. Questo componente è responsabile delle interazioni con le basi di dati, i sistemi di reportistica e le fonti esterne e governa le attività di *enrichment* (es. aggancio autori/progetti) guidati dall'utente, da servizi esterni o da processi semi-supervisionati.

Il *frontend* è sviluppato in *TypeScript* sul framework *Angular* e sulla libreria **SisinfoShell** del sistema informativo. Questo componente è responsabile delle interazioni con l'utente e permette di gestire i casi d'uso relativi alla navigazione e alla fruizione/gestione del dato.

6 UI/UX

L'applicazione web del sistema gestionale è stata sviluppata utilizzando moderne tecnologie web, in particolare è stato scelto il framework *Angular* arricchito con le seguenti librerie:

- PrimeNG – componenti UI;
- NgRx – architettura *push-based*;
- RxJs – gestione dei flussi dati all'interno dell'app.

7 Angular

Angular è un framework open-source molto popolare per lo sviluppo di applicazioni web. Tra i punti di forza di Angular possiamo elencare:

- **Architettura ben definita**: Angular offre un design architetturale ben definito che facilita lo sviluppo di applicazioni web complesse;
- **Rich UI/UX**: Angular include una vasta gamma di funzionalità per la creazione di interfacce utente interattive e dinamiche. Inoltre, può essere utilizzato insieme a librerie che forniscono componenti UI pronti, facili da inserire e da personalizzare;
- **Manutenibilità**: Angular promuove l'adozione di principi SOLID[10] che favoriscono la manutenibilità dell'applicazione;
- **Supporto**: solido supporto da parte di Google e vasta community.

7.1 Architettura dell'applicazione

L'applicazione "Prodotti della ricerca" è fortemente *data-centric*: tutte le funzionalità ruotano attorno alla disponibilità del dato "Prodotto". In quest'ottica, l'utilizzo di NgRx ed RxJs ha consentito lo sviluppo di un'applicazione interamente *push-based:* il dato è l'attore principale, che popola le viste ed è oggetto di manipolazione.

Nei seguenti paragrafi approfondiamo l'utilizzo della strategia *push-based*.

7.2 Approccio pull-based

L'approccio tradizionale allo sviluppo di applicazioni web è *pull-based*. Le viste chiamano esplicitamente i metodi per recuperare i dati; se i dati cambiano, le viste devono richiedere il reload dei dati.

Il collegamento tra viste e dati, mantenuto per tutto il ciclo di vita dell'applicazione, prende il nome di *long-term data flow*. In un contesto moderno di applicazioni asincrone e *rich user experience*, librerie come RxJs ed in generale la programmazione reattiva sono di grande aiuto: alla base della programmazione reattiva ci sono gli *Observables*, oggetti che consentono la gestione di un flusso di dati ed eventi sincrono/asincrono nel tempo.

In un'architettura pull-based si possono utilizzare gli Observables per recuperare i dati dal backend e distribuirli alle viste/servizi interessati, ma sono tipicamente utilizzati come *stream* temporanei per eseguire una singola chiamata al backend e distrutti dopo l'uso.

L'approccio pull-based ha in generale diverse criticità:

- **manutenibilità**: l'implementatione del long-term data flow richiede sistemi di polling e callback che tendono a complicare l'architettura dell'applicazione e di conseguenza la sua manutenibilità;

[10] https://en.wikipedia.org/wiki/SOLID

- **prestazioni**: la strategia pull-based richiede l'utilizzo del *Change Detection* (ciclo di rilevamento delle modifiche) da parte di Angular, in base al quale si decide quali viste devono essere aggiornate; il Change Detection ha notoriamente un impatto prestazionale non trascurabile;
- **data governance**: se i dati sono condivisi tra diverse viste, nasce il problema di dove salvare i dati, come distribuirli tra i vari interessati e come gestirne l'aggiornamento.

7.3 Approccio push-based

Un'architettura push-based in Angular supera le criticità dell'approccio pull-based:

- **prestazioni**: l'uso di servizi push-based che forniscono dati solo attraverso streams incoraggia lo sviluppo di applicazioni composte da *Reactive Views*: viste passive che si aggiornano solo quando i dati arrivano tramite uno stream *push*. Queste componenti sono altamente performanti perché consentono di disabilitare il meccanismo di *Change Detection* di Angular;
- **data dovernance e manutenibilità**: con l'approccio pull-based la complessità aumenta, soprattutto quando più viste richiedono gli stessi dati. In un'architettura push-based è possibile utilizzare un pattern *Facade* per centralizzare la logica e gestire le dipendenze in modo più ordinato. Questo rende l'applicazione più facile da comprendere e manutenere;
- **long-term data flow "by design":** l'uso di stream RxJS a lunga durata implementa un processo di invio automatico delle modifiche ai dati a tutti gli interessati. Le componenti si limitano a sottoscrivere gli stream di dati desiderati e quando la sorgente dati cambia, le modifiche vengono inviate automaticamente attraverso lo stream a tutti i sottoscrittori.

In conclusione, un'architettura *push-based* in Angular offre vantaggi in termini di reattività delle view, gestione delle dipendenze e notifica automatica delle modifiche ai dati. Inoltre, utilizzando librerie come NgRx ed RxJS, è possibile creare soluzioni *push-based* eleganti e performanti.

8 Opportunità di ricerca: AI e ML

Un filone R&D molto promettente che può potenziare ed arricchire le funzionalità di un sistema CRIS è l'utilizzo di applicazioni di Machine Learning (ML).

In generale, il ML può migliorare l'efficienza e la precisione di un sistema CRIS su diversi fronti:

- classificazione e categorizzazione degli articoli: il ML può essere utilizzato per classificare e categorizzare gli articoli scientifici in base al loro contenuto, al loro argomento o al loro campo di ricerca. Questo caso d'uso può aiutare a **organizzare** meglio la produzione scientifica;

- identificazione di errori e problematiche nei dati: il ML può essere utilizzato per identificare potenziali errori nei dati, come ad esempio dati mancanti o anomali. Questo caso d'uso può aiutare a migliorare la **qualità** del sistema CRIS;
- analisi dei dati: il ML può essere utilizzato per analizzare grandi quantità di dati e per identificare *trend*, correlazioni e relazioni nascoste tra variabili. Questo caso d'uso costituisce un valore aggiunto del sistema CRIS che può promuoverne la **diffusione**;
- predizione dei trend futuri: il ML può essere utilizzato per predire *trend* futuri sulla base dei dati esistenti e delle tendenze attuali. Ciò può aiutare a individuare **aree di ricerca promettenti**.

L'ambito di applicabilità è chiaramente molto vasto, la sfida iniziale è quella di porre le basi per esplorare tutte le potenzialità.

I primi casi d'uso che il ML può coprire sono:

- riconoscimento automatico delle sigle di progetto sulla base dei metadati di un prodotto scientifico;
- riconoscimento automatico degli autori e delle affiliazioni;
- riconoscimento automatico delle relazioni tra preprint e VOR.

L'obiettivo è quello di individuare i modelli di ML ed i dataset di training per soddisfare questi casi d'uso, a partire dai quali si potranno costruire modelli più complessi per risolvere problematiche più ambiziose.

In termini architetturali, il servizio di ML è un applicativo parallelo al sistema di gestione della produzione scientifica: una black-box che espone API di definizione e di training dei modelli ed API di predizione che sfruttano i modelli addestrati per i casi d'uso specifici.

La scelta di mantenere il servizio di ML separato dall'applicativo principale è indirizzata da due fattori:

- il servizio di ML ha esigenze architetturali differenti, a partire dalla necessità di ricorrere ad HW dotato di GPUs;
- si vuole sviluppare un servizio di ML in una prospettiva di generalizzazione, dove il progetto di gestione della produzione scientifica sarà il primo ed il principale caso d'uso, ma ulteriori scenari potranno essere esplorati indipendentemente.