ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA

CORSO DI LAUREA IN
INGEGNERIA E SCIENZE INFORMATICHE

Riprogettazione di un'Applicazione Web per l'Analisi delle Pubblicazioni Scientifiche

Tesi di laurea in Tecnologie Web

Relatore:
Prof. Giovanni Delnevo

Presentata da:
Morellini Matteo

Correlatori:
Prof. Silvia Mirri
Dr. Manuel Andruccioli

Sessione Marzo 2026
Anno Accademico 2024-2025

A chi si sente secondo, ma continua a camminare con coraggio verso ciò che desidera. A chi cade, si rialza, e senza clamore trova la propria strada. ...

Introduzione

La valutazione della produzione scientifica si colloca oggi in un contesto caratterizzato da un'elevata disponibilità di dati bibliografici e bibliometrici, distribuiti su piattaforme differenti e governati da modelli di accesso, copertura e rappresentazione eterogenei. Database generalisti come Scopus e Web of Science forniscono metriche quantitative e profili autore; portali specialistici come DBLP garantiscono una copertura accurata delle pubblicazioni in ambito informatico; strumenti quali SCImago e CORE offrono indicatori qualitativi sulle sedi editoriali.

Questa pluralità di fonti rappresenta al tempo stesso una risorsa e una criticità: consente analisi articolate e multidimensionali, ma introduce una frammentazione dei dati, delle interfacce e della semantica che rende complessa la costruzione di un quadro coerente e sintetico del profilo di un ricercatore. Per ottenere una vista completa, l'utente è spesso costretto a interrogare più portali, ripetere ricerche simili su interfacce differenti e riconciliare manualmente risultati non sempre allineati, con un costo operativo e cognitivo significativo e un rischio non trascurabile di errori e incoerenze.

In questo scenario si inserisce DocRanks, progetto nato per offrire una vista unificata dei dati provenienti da fonti bibliometriche eterogenee, inizialmente realizzato come applicazione web monolitica in PHP con rendering server-side. Pur risultando funzionale, la versione originaria presentava limiti strutturali in termini di modularità, separazione delle responsabilità, facilità di integrazione con nuove API e capacità di supportare funzionalità interattive avanzate, rendendo complessa l'evoluzione del sistema rispetto alle

esigenze emerse nel dominio bibliometrico.

L'obiettivo della presente tesi è progettare e realizzare DocRanks 2.0, una nuova versione del sistema basata su un'architettura a componenti composta da backend API-centrico in Python, *Single Page Application* (SPA) in React e persistenza locale su database relazionale, in grado di integrare in modo coerente Scopus, DBLP, SCImago e CORE. L'intento è duplice: superare i limiti dell'implementazione originaria, sia sul piano architetturale sia su quello dell'esperienza d'uso, e costruire un'infrastruttura solida, estendibile e coerente con i requisiti di tracciabilità, verificabilità e gestione dell'incertezza che caratterizzano l'integrazione di fonti bibliometriche.

Il contributo principale del lavoro consiste nella definizione di un modello interno coerente per la rappresentazione congiunta di autori, pubblicazioni e venue, nell'implementazione di una pipeline di normalizzazione e matching delle sedi editoriali, nella progettazione di un backend strutturato in layer distinti (*routers*, servizi, *repositories*) e nello sviluppo di una SPA orientata alla consultazione progressiva dei dati. A questi elementi si aggiunge l'integrazione di una Chat AI con accesso controllato al database tramite *function calling*, concepita come supporto interpretativo alla lettura dei dati piuttosto che come semplice interfaccia alternativa, e l'adozione di una persistenza locale come *single source of truth* per garantire consultazioni ripetibili e verificabili.

Il volume di tesi è organizzato nei seguenti capitoli:

- **Capitolo 1** – analizza il contesto bibliometrico e tecnologico in cui si colloca DocRanks, descrivendo i principali database (Scopus, Web of Science, DBLP), gli strumenti di valutazione qualitativa (SCImago, CORE), le metriche bibliometriche più rilevanti e le criticità derivanti dalla frammentazione dei dati, delle interfacce e della semantica.

- **Capitolo 2** – formalizza il problema affrontato da DocRanks 2.0 in termini di integrazione e coerenza dei dati, definisce obiettivi funzionali e non funzionali, identifica stakeholder e casi d'uso e presenta il design concettuale della soluzione proposta, con particolare attenzione

all'architettura logica, ai flussi principali e al modello concettuale dei dati.

- **Capitolo 3** – descrive la realizzazione di DocRanks 2.0, illustrando la struttura del repository, il backend Python basato su FastAPI, la SPA sviluppata con React, la pipeline ETL per l'importazione e la normalizzazione delle fonti bibliometriche e l'integrazione della Chat AI, mettendo in evidenza le principali scelte progettuali e le loro implicazioni in termini di manutenibilità, estendibilità e qualità del codice.

Nel complesso, la tesi propone un'evoluzione architetturale che trasforma DocRanks da applicazione monolitica a sistema modulare e orientato ai servizi, capace di integrare fonti eterogenee mantenendo coerenza interna e offrendo un'esperienza di consultazione uniforme e verificabile, in linea con le convenzioni tipiche dei documenti scientifici della disciplina.

Indice

Elenco delle figure

Capitolo 1

Contesto di Progetto

La valutazione della produzione scientifica si basa oggi su un insieme eterogeneo di banche dati e sistemi di classificazione, ciascuno progettato e gestito in modo indipendente. Questa frammentazione costituisce una delle principali difficoltà da tenere in considerazione per ricercatori, gruppi di ricerca e istituzioni accademiche, che devono consultare più fonti per ottenere un quadro completo della qualità di una pubblicazione o del profilo bibliometrico di un autore.

Per esempio, il valore di FWCI (*Field-Weighted Citation Impact*) è disponibile su Scopus, il quartile di una rivista va recuperato su SCImago, altre metriche come CiteScore o SNIP provengono ancora da Scopus, mentre per l'elenco strutturato delle pubblicazioni informatiche, che consente di avere un quadro più completo, è spesso necessario interrogare DBLP. Questo processo, se svolto manualmente, richiede tempo, competenze specifiche e comporta un rischio non trascurabile di errori o incoerenze durante la trascrizione dei dati.

In questa prospettiva, il presente capitolo analizza i principali strumenti dedicati alla catalogazione e alla valutazione della ricerca scientifica, con particolare riferimento alle piattaforme più utilizzate nell'ambito informatico. Verranno descritte le loro caratteristiche distintive, i criteri adottati per classificare riviste e conferenze, le metriche bibliometriche più rilevanti e le criticità che derivano dall'assenza di un modello unificato di rappresentazione

dei dati.

Il contesto tecnologico del progetto. Una prima versione di DocRanks era stata realizzata come applicazione web in PHP. Nonostante fosse funzionale, essa presentava i limiti tipici delle applicazioni monolitiche e server-side accennati in precedenza. La struttura del codice rendeva inoltre complessa l'integrazione con nuove API e l'introduzione di componenti interattivi o visualizzazioni avanzate.

L'obiettivo di DocRanks era però chiaro: offrire una vista unificata dei dati provenienti da Scopus, DBLP, SCImago e CORE. Senza un'architettura client-centrica, tuttavia, il sistema non riusciva a esprimere appieno questo potenziale. Si è reso quindi necessario un ripensamento radicale, che ha portato alla progettazione di DocRanks 2.0.

La trasformazione in SPA. L'adozione di una *Single Page Application* (SPA) è emersa come soluzione per superare i limiti dell'implementazione originaria.[1] Una SPA permette brevemente di:

- gestire la navigazione senza dover ricaricare, ad ogni iterazione con qualsiasi funzionalità presenti, la pagina;
- mantenere lo stato applicativo lato client, con benefici in termini di performance e coerenza;
- interagire in modo naturale con un backend API-centrico;
- integrare visualizzazioni dinamiche, tabelle reattive e funzioni avanzate di ricerca;
- offrire un'esperienza d'uso moderna, comparabile alle principali piattaforme bibliometriche.

La migrazione verso una SPA ha richiesto una revisione completa dell'architettura: refactoring dei processi di comunicazione con le API, progettazione di un backend modulare e sviluppo di un frontend basato su React, Vite e TailwindCSS. Il risultato è una piattaforma più robusta, scalabile e

predisposta per future estensioni, tra cui l'integrazione della *Chat AI* che arricchisce ulteriormente l'esperienza utente.

In sintesi, il contesto del progetto non riguarda soltanto la comprensione dei database bibliografici e delle metriche di valutazione, ma soprattutto la necessità di progettare un'infrastruttura applicativa capace di integrarli in modo efficace e coerente. Il passaggio da DocRanks alla versione 2.0 — dalla vecchia applicazione PHP a una SPA moderna — rappresenta quindi un'evoluzione necessaria per costruire uno strumento più versatile, efficiente e adatto a operare in un ecosistema bibliometrico complesso ed in continua crescita.

1.1 Evoluzione della valutazione bibliometrica e ruolo nelle decisioni accademiche

Negli ultimi decenni, l'uso di metriche bibliometriche si è progressivamente consolidato come supporto ai processi di valutazione della ricerca. Indicatori quantitativi (come numero di citazioni, H-index o metriche normalizzate) e classificazioni qualitative (come quartili di rivista o ranking di conferenza) vengono impiegati per confrontare profili scientifici, monitorare la produttività, identificare sedi editoriali di riferimento e valutare la visibilità di una linea di ricerca.

Questo fenomeno è legato a più fattori. Da un lato, l'aumento del volume complessivo di pubblicazioni rende difficile un'analisi puramente qualitativa basata su lettura e revisione manuale. Dall'altro, istituzioni e enti di finanziamento necessitano di strumenti di sintesi che permettano valutazioni comparabili in tempi ragionevoli. In tale contesto, le metriche fungono da "proxy" dell'impatto, pur con limiti noti e discussi nella letteratura di settore.

È importante sottolineare che le metriche non possono essere interpretate come misure assolute di "qualità" scientifica. Citazioni e indicatori deri-

vati sono influenzati da molte variabili: disciplina, dimensione della comunità, abitudini citazionali, tipologia di pubblicazione (rivista vs conferenza), età dei lavori e anche fattori geografici o linguistici. Per questo motivo è prassi consolidata affiancare più indicatori tra loro, ricorrendo a metriche normalizzate e a sistemi di classificazione che tentano di ridurre i bias disciplinari.

In ambito informatico questa esigenza è ancora più evidente: molte aree della computer science privilegiano le conferenze come sede primaria di pubblicazione e diffusione dei risultati, mentre altri settori scientifici danno maggiore peso alle riviste. Ne consegue che un sistema di valutazione efficace deve essere in grado di combinare correttamente fonti diverse e rappresentare con chiarezza la natura delle pubblicazioni analizzate.

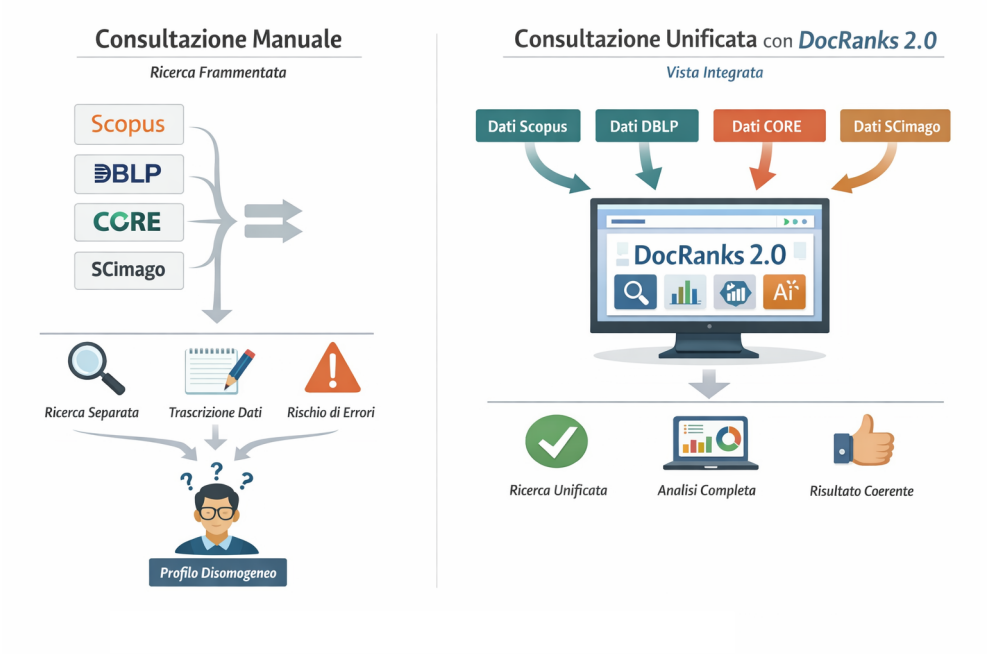

Figura 1.1: Confronto concettuale tra consultazione manuale di più portali e consultazione unificata in DocRanks 2.0.

DocRanks 2.0 si inserisce in questo scenario con l'obiettivo di ridurre la complessità operativa per l'utente, facilitando l'accesso e l'interpretazione di informazioni distribuite su più fonti. In particolare, l'idea di fondo è

rendere più immediata la costruzione di un quadro complessivo, in cui dati bibliografici e indicatori qualitativi siano consultabili in modo coerente e con un percorso di navigazione uniforme.

1.2 Database bibliografici

I database bibliografici (DB) sono enormi raccolte digitali di riferimenti a pubblicazioni scientifiche. In essi è possibile trovare informazioni strutturate su articoli di riviste, atti di conferenze, libri, capitoli e molte altre tipologie di contributi. Ogni pubblicazione è descritta da metadati quali titolo, autori, affiliazioni, anno di pubblicazione e identificativi standard (ad esempio il DOI).

Nel contesto della ricerca scientifica moderna, i DB rappresentano il principale punto di accesso alla letteratura: consentono di reperire rapidamente i lavori pertinenti a un argomento, ma anche di calcolare indicatori quantitativi (numero di citazioni, H-index, metriche normalizzate) utili per analizzare l'impatto scientifico di autori, riviste e conferenze.

L'utilizzo dei DB si è progressivamente esteso oltre la semplice consultazione: oggi essi sono strumenti centrali per monitorare la propria produzione, individuare collaborazioni, seguire l'evoluzione delle aree di ricerca e valutare la qualità delle sedi editoriali. Al tempo stesso, presentano importanti differenze in termini di copertura disciplinare, modello di accesso e tipologia di informazioni offerte.

Nei paragrafi che seguono vengono presentati i principali database utilizzati nel progetto DocRanks, evidenziandone il ruolo e i limiti in relazione all'analisi bibliometrica.

Tipologie di pubblicazione e specificità dell'ambito informatico

La valutazione bibliometrica dipende in modo significativo dalla tipologia di pubblicazione. In molte discipline la rivista rappresenta il canale principale di disseminazione, con tempi di revisione e pubblicazione spesso più lunghi e una forte enfasi sulla stabilità dei risultati. In informatica, invece, numerose aree adottano un modello differente: le conferenze internazionali costituiscono spesso la sede primaria di pubblicazione, con processi di selezione competitivi e cicli annuali che favoriscono una rapida circolazione delle innovazioni.

Questa peculiarità influenza direttamente la lettura delle metriche. Ad esempio, il confronto tra autori basato esclusivamente su indicatori legati alle riviste può risultare penalizzante per chi pubblica prevalentemente su conferenze. Analogamente, la copertura di database generalisti può non riflettere in modo uniforme l'importanza delle venue conferenziali, rendendo necessario integrare fonti specializzate e strumenti qualitativi dedicati.

Nel contesto di DocRanks 2.0, tale distinzione è rilevante perché motiva l'uso congiunto di banche dati bibliografiche (per l'elenco e i metadati) e ranking qualitativi (per contestualizzare la sede), così da rappresentare in modo più fedele la produzione scientifica nel dominio della computer science.

Differenze tra copertura, granularità e modello di accesso

Sebbene i database bibliografici condividano l'obiettivo generale di indicizzare la letteratura scientifica, differiscono notevolmente per copertura e criteri di selezione. Alcuni database privilegiano la qualità e l'affidabilità dei metadati tramite processi editoriali e comitati di selezione, mentre altri puntano a una copertura più ampia o a una forte specializzazione disciplinare.

Un secondo aspetto rilevante è la granularità dei dati: oltre ai metadati di base, alcune piattaforme includono profili autore, affiliazioni istituziona-

li, relazioni di citazione e metriche derivate, rendendo possibile non solo la ricerca bibliografica ma anche analisi bibliometriche avanzate.

Infine, il modello di accesso può essere chiuso (tramite abbonamenti e licenze) oppure aperto (open data o API liberamente consultabili). Questo influenza direttamente la facilità con cui un sistema terzo può integrare i dati. DocRanks 2.0, dovendo combinare più fonti, tiene conto di questi vincoli sia nella fase di importazione sia nella presentazione all'utente.

Coerenza e tracciabilità dei dati tra fonti

Quando si combinano informazioni provenienti da fonti differenti, due requisiti diventano centrali: la *coerenza* dei dati e la loro *tracciabilità*. Per coerenza si intende la capacità di presentare valori e metadati senza contraddizioni evidenti (ad esempio evitare duplicazioni della stessa pubblicazione o disallineamenti tra titolo, anno e venue). Per tracciabilità si intende invece la possibilità di ricondurre ogni informazione alla fonte originaria, così da interpretare correttamente eventuali differenze tra piattaforme.

In ambito bibliometrico, discrepanze tra conteggi e metriche non sono necessariamente errori: possono derivare da differenze di copertura, criteri di indicizzazione o finestre temporali adottate dai vari provider. Una vista unificata deve quindi puntare non solo a "sommarizzare", ma anche a rendere il dato leggibile e verificabile, riducendo ambiguità e migliorando la fiducia dell'utente nei risultati mostrati.

1.3 Scopus

Scopus è un database bibliografico sviluppato da Elsevier e introdotto nel 2004 come alternativa moderna a Web of Science. È una delle piattaforme più estese per l'indicizzazione delle pubblicazioni scientifiche, con una copertura disciplinare ampia che include riviste, atti di conferenze, libri e serie editoriali aggiornati quotidianamente ma soprattutto scelti con cura.

Gestione dei contenuti

L'indicizzazione in Scopus combina procedure automatiche e controlli umani. Algoritmi proprietari estraggono metadati, riconoscono le citazioni e associano gli autori alle loro pubblicazioni.

La selezione delle riviste e delle conferenze è supervisionata dal *Content Selection and Advisory Board* (CSAB), che verifica continuità editoriale, qualità del processo di revisione e rilevanza scientifica.

Per ridurre gli errori di disambiguazione degli autori, Scopus utilizza insiemi di riferimento e strumenti dedicati, come l'*Author Feedback Wizard*, tramite cui i ricercatori possono richiedere la correzione di profili duplicati o incompleti. La copertura delle citazioni è completa a partire dagli anni Novanta, con una retrospettiva storica che arriva fino alla fine del XVIII secolo.

Scopus mette inoltre a disposizione API ufficiali, basate su architettura REST,[2] che consentono l'integrazione programmatica con sistemi esterni come DocRanks.[3]

Metriche principali

Oltre alla funzione di catalogo, Scopus fornisce diverse metriche bibliometriche, tra cui:

- **numero di documenti e citazioni** associati ad un autore;
- **H-index**, calcolato sui dati interni alla piattaforma;
- **Field-Weighted Citation Impact (FWCI)**, che confronta le citazioni ricevute da un articolo con quelle attese per articoli simili;
- **CiteScore**, indicatore basato sulle citazioni ricevute in una finestra di quattro anni;
- **SNIP** (*Source Normalized Impact per Paper*), che normalizza l'impatto delle citazioni rispetto alle abitudini citazionali del settore.

Queste metriche costituiscono la base quantitativa da cui DocRanks ricava molte delle informazioni mostrate nella SPA.

Limitazioni

Nel settore informatico la copertura delle conferenze è meno completa rispetto a quella delle riviste e, soprattutto, Scopus non fornisce una misura diretta della qualità delle venue conferenziali. Per questo DocRanks integra i dati Scopus con il ranking CORE.

Scopus non espone inoltre i quartili delle riviste, calcolati da SCImago a partire dall'indicatore SJR, e le API sono soggette a limiti di utilizzo e a vincoli di licenza. Questi aspetti richiedono una gestione attenta delle interrogazioni e la combinazione dei dati con altre fonti.

Considerazioni sull'uso in un sistema integrato

Nel momento in cui Scopus viene utilizzato come fonte primaria in un sistema di integrazione, emergono alcuni aspetti pratici legati alla struttura e alle modalità di accesso ai dati. In particolare, è necessario garantire la coerenza tra gli identificativi forniti dalla piattaforma (ad esempio *Author ID* e identificativi delle pubblicazioni) e le informazioni provenienti da altre fonti bibliografiche.

Un ulteriore elemento da considerare è che i dati messi a disposizione dalle API possono risultare distribuiti su più endpoint e, in alcuni casi, incompleti o soggetti a vincoli di accesso. Questo richiede una progettazione attenta del flusso di acquisizione delle informazioni, al fine di presentare all'utente finale un insieme di dati coerente e interpretabile.

Il sistema privilegia la presentazione di dati consolidati e chiaramente attribuibili alla fonte di origine, evitando di basarsi su interrogazioni dirette e ripetute ad ogni interazione dell'utente.

1.3.1 DBLP

DBLP (*Digital Bibliography & Library Project*) è un database bibliografico specializzato nell'informatica. Nato come iniziativa accademica, è oggi una delle principali piattaforme open data dedicate alla catalogazione delle pubblicazioni informatiche, con particolare enfasi su conferenze, workshop e riviste del settore.

Gestione e caratteristiche

I contenuti di DBLP sono raccolti a partire da informazioni fornite da editori e organizzatori di eventi scientifici e vengono sottoposti a controlli di coerenza, normalizzazione e disambiguazione degli autori. Il risultato è una base di dati accurata e ben strutturata, spesso utilizzata come riferimento per studi bibliometrici in ambito computer science.

A differenza di Scopus e Web of Science, DBLP non fornisce metriche di citazione o indicatori di qualità; il suo punto di forza è la chiarezza con cui rappresenta il profilo autoriale e l'origine delle pubblicazioni (rivista, conferenza, workshop).

Il portale mette tuttavia a disposizione meccanismi di interrogazione programmatica tramite endpoint HTTP che restituiscono risultati in formati strutturati, come XML e JSON, oltre a dataset completi scaricabili pubblicamente.[4] Questa modalità di accesso, pur non configurandosi come una API REST formale, consente l'integrazione dei dati DBLP in sistemi esterni come DocRanks.

Limiti

L'assenza di dati sulle citazioni e di metriche qualitative richiede che DBLP venga affiancato da altre fonti quando si vogliono valutare l'impatto o il prestigio delle pubblicazioni. Nonostante ciò, per la comunità informatica rimane uno strumento essenziale grazie all'accuratezza dei metadati e alla copertura estesa delle conferenze.

Valore aggiunto per il dominio informatico

In informatica, l'identificazione della sede di pubblicazione è particolarmente importante perché molte conferenze rappresentano il canale principale di disseminazione dei risultati. DBLP, attraverso la sua struttura, consente una distinzione molto chiara tra tipologie di venue (conferenze principali, workshop, riviste, volumi di atti), facilitando l'analisi delle traiettorie di pubblicazione di un autore.

Questo aspetto risulta utile soprattutto quando un database generalista presenta lacune nella copertura delle conferenze o quando è necessario disporre di un elenco bibliografico ordinato e coerente da arricchire successivamente con metriche e classificazioni.

1.3.2 Web of Science

Web of Science (WoS), oggi gestito da Clarivate Analytics, è uno dei database bibliografici più consolidati e influenti. È multidisciplinare e copre un'ampia gamma di settori scientifici tramite diverse collezioni, tra cui:

- **SCIE** (Science Citation Index Expanded),
- **SSCI** (Social Sciences Citation Index),
- **AHCI** (Arts & Humanities Citation Index),
- **ESCI** (Emerging Sources Citation Index),
- **CPCI** (Conference Proceedings Citation Index).

L'inclusione in WoS è subordinata a un processo di selezione rigoroso, che valuta qualità editoriale, peer review e rilevanza internazionale delle riviste.

Tra le metriche offerte si trovano:

- numero di pubblicazioni e citazioni per autore;
- **H-index** calcolato sulla base dei dati WoS;

- **Impact Factor (IF)**, basato sulle citazioni a due anni;
- **Journal Citation Indicator (JCI)**, che normalizza le citazioni per disciplina;
- strumenti di analisi della *citation network*.

WoS è ampiamente utilizzato nei processi istituzionali di valutazione, ma presenta alcune criticità: copertura limitata delle conferenze informatiche, costi di accesso elevati e possibile bias linguistico e geografico. Per questi motivi, nel progetto DocRanks viene utilizzato principalmente come riferimento concettuale, mentre i dati operativi provengono da Scopus, DBLP, CORE e SCImago.

Considerazioni sull'eterogeneità dei database

Il confronto tra WoS e Scopus evidenzia un aspetto cruciale: lo stesso autore può avere valori diversi di H-index e citazioni a seconda della fonte consultata, a causa di differenze di copertura e criteri di indicizzazione. Questo rende importante, quando si integrano dati, dichiarare la fonte delle metriche e ridurre al minimo ambiguità nella presentazione.

DocRanks 2.0 adotta quindi un approccio orientato alla chiarezza: le metriche sono presentate in modo coerente e riconducibile alla piattaforma da cui provengono, consentendo all'utente di interpretarle correttamente.

1.4 Strumenti per la valutazione qualitativa

Oltre ai database bibliografici generalisti, la valutazione scientifica si avvale di strumenti dedicati alla classificazione qualitativa di riviste e conferenze. In questa categoria rientrano il ranking CORE, orientato alle conferenze di area informatica, e il portale SCImago Journal & Country Rank, focalizzato sulle riviste indicizzate in Scopus.

Questi sistemi non mirano a elencare tutte le pubblicazioni, ma a sintetizzare tramite scale ordinali o indicatori numerici il prestigio e l'impatto delle

sedi editoriali. Nel progetto DocRanks essi arricchiscono i dati provenienti dai DB con informazioni qualitative sulle venue.

1.4.1 CORE

CORE (*Computing Research and Education Association of Australasia*) mantiene un sistema di classificazione delle conferenze nell'area dell'informatica e delle tecnologie dell'informazione. Il portale CORE Conference Rankings è utilizzato perANKING CORE è utilizzato per valutare in modo sintetico il prestigio e la selettività delle sedi conferenziali.

CORE non indicizza i singoli articoli né fornisce metriche di citazione: assegna invece a ciascuna conferenza una classe di qualità (**A***, **A**, **B**, **C**, dove la prima classificazione rappresenta una categoria più prestigiosa e di maggior rilevanza, scendendo l'importanza diminuisce), sulla base di informazioni raccolte da società scientifiche, comitati organizzatori ed esperti di settore. I criteri considerati includono reputazione storica, tassi di accettazione, qualità del peer review e continuità dell'evento.

Nel contesto di DocRanks, il ranking CORE è utilizzato per annotare gli atti di conferenza con un indicatore qualitativo, che affianca il numero di citazioni e consente di distinguere rapidamente tra venue di fascia alta e venue meno prestigiose.

Tra i limiti del sistema si segnalano la copertura concentrata quasi esclusivamente sull'informatica, l'aggiornamento non sempre tempestivo e l'assenza di un'API pubblica stabile, che rende necessarie procedure di importazione e normalizzazione dei dati.

Interpretazione e limiti d'uso

Le classificazioni qualitative, pur essendo molto utili per una valutazione rapida, devono essere interpretate con cautela. L'assegnazione di una classe può dipendere da criteri non uniformi tra le diverse aree dell'informatica e può risentire della variabilità nel tempo (ad esempio per conferenze emergenti o per cambiamenti nel processo di selezione dei contributi).

DocRanks 2.0 utilizza CORE come informazione aggiuntiva per contestualizzare i risultati, ma la presenta come indicatore qualitativo complementare e non come metrica assoluta.

1.4.2 SCImago

SCImago Journal & Country Rank (SJCR) è un portale pubblico che fornisce indicatori bibliometrici per riviste e paesi, calcolati a partire dai dati di citazione di Scopus. Il gruppo di ricerca SCImago Lab ha sviluppato l'indicatore *SCImago Journal Rank* (SJR) come alternativa libera e trasparente ad altre metriche proprietarie.

Per ciascuna rivista il portale mette a disposizione, tra gli altri:

- il valore di **SJR**, che misura la "prestigiosità" della rivista pesando maggiormente le citazioni provenienti da sedi autorevoli;
- l'**H-index** di rivista;
- il numero medio di citazioni per documento (**Cites per Doc**);
- la suddivisione in quartili (**Q1**–**Q4**) all'interno di ciascuna categoria disciplinare.

Nel progetto DocRanks, SCImago è utilizzato per associare a ogni articolo di rivista il quartile e il valore di SJR della relativa sede editoriale, permettendo così di distinguere, ad esempio, tra pubblicazioni in riviste Q1 e Q3.

Anche in questo caso esistono alcuni limiti: la dipendenza dai dati Scopus, l'aggiornamento tipicamente annuale e l'assenza di un'API completa e ufficiale. L'integrazione con DocRanks avviene quindi tramite dataset esportati e successiva normalizzazione.

Quartili e multi-categorizzazione

Un aspetto importante di SCImago è che una rivista può appartenere a più categorie disciplinari. Questo implica che una stessa rivista possa avere

quartili diversi a seconda della categoria considerata. In un sistema integrato è quindi necessario definire una strategia di rappresentazione (ad esempio scegliendo la categoria prevalente o mostrando più categorie). Nel contesto di DocRanks 2.0, questa complessità viene gestita con l'obiettivo di fornire una vista chiara e utile all'utente, senza sovraccaricare l'interfaccia con informazioni ridondanti.

1.5 Metriche bibliometriche principali

Le piattaforme descritte mettono a disposizione un'ampia gamma di indicatori quantitativi. In questa sezione vengono presentate le metriche più rilevanti per DocRanks 2.0, organizzate in tre gruppi: produttività e impatto individuale, metriche normalizzate e indicatori di prestigio delle sedi editoriali.

Numero di documenti e citazioni

Il conteggio dei documenti associati a un autore e delle citazioni ricevute rappresenta il punto di partenza di qualunque analisi bibliometrica. Il *numero di documenti* descrive la produttività, mentre il *numero di citazioni* offre una prima misura della visibilità delle pubblicazioni nel tempo. Nel profilo autore di DocRanks questi valori sono presentati come informazione di base, su cui si innestano metriche più sofisticate.

È opportuno notare che i conteggi possono variare tra piattaforme, a causa di differenze di copertura e criteri di indicizzazione. Questo vale sia per il totale delle citazioni sia per l'elenco delle pubblicazioni attribuite a un autore.

H-index

L'H-index, proposto da Hirsch nel 2005, combina produttività e impatto: un autore ha H-index pari a h se ha pubblicato almeno h articoli che hanno

ricevuto ciascuno almeno h citazioni. Sia Scopus sia Web of Science calcolano tale indice sui propri dati; in DocRanks viene utilizzato il valore fornito da Scopus per caratterizzare sinteticamente il profilo del ricercatore.

Tra i limiti dell'H-index si evidenziano la dipendenza dalla durata della carriera (favorisce autori con pubblicazioni più datate), l'insensibilità a poche pubblicazioni estremamente citate e l'impossibilità di confrontare correttamente discipline con pratiche citazionali molto diverse.

Metriche normalizzate: FWCI, SNIP e JCI

Per confrontare autori di discipline diverse o con abitudini citazionali differenti sono state introdotte metriche normalizzate:

- il **Field-Weighted Citation Impact (FWCI)**, che misura il rapporto tra citazioni ricevute e citazioni attese per articoli simili (per anno, tipologia e area disciplinare);
- lo **SNIP**, che normalizza l'impatto delle citazioni di una rivista rispetto alle consuetudini citazionali del campo scientifico;
- il **Journal Citation Indicator (JCI)**, concettualmente analogo al FWCI ma basato sui dati di Web of Science.

In DocRanks, FWCI e SNIP vengono utilizzati per interpretare le citazioni di un autore nel contesto del proprio settore, riducendo i bias disciplinari.

Metriche di rivista: Impact Factor, CiteScore e SJR

Per valutare le sedi editoriali si utilizzano indicatori specifici:

- l'**Impact Factor (IF)**, calcolato da Web of Science sulle citazioni a due anni;
- il **CiteScore**, fornito da Scopus con finestra di quattro anni;

- lo **SJR** di SCImago, che utilizza un modello a reti di citazione attribuendo peso diverso alle fonti citanti.

In DocRanks l'attenzione è focalizzata soprattutto su SJR, utilizzato in combinazione con i quartili per caratterizzare il posizionamento delle riviste.

Quartili di rivista (Q1–Q4)

Sulla base dell'indicatore SJR, SCImago suddivide le riviste in quartili (**Q1**, **Q2**, **Q3**, **Q4**) all'interno di ciascuna categoria disciplinare. Le riviste Q1 appartengono al 25% con SJR più elevato; Q4 raccoglie il 25% con valori più bassi. DocRanks integra questi dati per mostrare, accanto a ogni articolo, la fascia qualitativa della rivista di pubblicazione.

Classificazione delle conferenze: ranking CORE

Nel caso delle conferenze, soprattutto in informatica, il numero di citazioni non è sempre sufficiente a rappresentare la qualità della sede. Il ranking CORE colma questa lacuna assegnando a ciascuna conferenza una classe qualitativa (**A***, **A**, **B**, **C**). In DocRanks tale informazione è visualizzata accanto agli atti di conferenza, permettendo confronti immediati tra sedi diverse.

Rilevanza delle metriche per DocRanks 2.0

DocRanks 2.0 integra queste metriche in un quadro coerente che combina:

- indicatori di produttività e impatto individuale (documenti, citazioni, H-index);
- metriche normalizzate per disciplina (FWCI, SNIP);
- indicatori di prestigio delle sedi editoriali (SJR, quartili SCImago, ranking CORE).

L'obiettivo non è sostituire i singoli portali, ma offrire una vista sintetica, unificata e navigabile che permetta di interpretare la produzione scientifica di un autore in modo più completo e bilanciato.

1.6 Vista unificata di DocRanks

Le differenze tra i portali bibliometrici non riguardano solo i dati disponibili, ma anche le interfacce con cui tali dati vengono presentati.

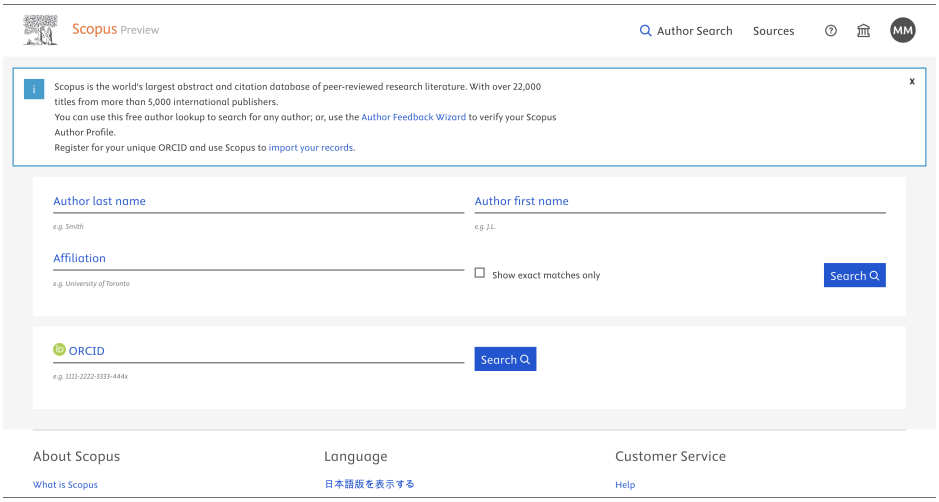

Figura 1.2: Interfaccia della ricerca di un autore di Scopus.

Mentre le maschere di ricerca di Scopus e Web of Science si distinguono per un'estrema linearità, le interfacce dedicate ai profili autore presentano una notevole densità funzionale. Tuttavia, la ricchezza di strumenti analitici, grafici e opzioni di esportazione si traduce spesso in un'architettura informativa complessa, frammentata su molteplici livelli di navigazione. Per ricostruire il profilo di un autore è necessario passare da una sezione all'altra: elenco delle pubblicazioni, citazioni, metriche, informazioni sulla rivista o sulla conferenza. L'utente deve quindi comporre mentalmente il quadro complessivo.

DBLP adotta un approccio opposto: interfaccia minimale, focalizzata sulle liste di pubblicazioni e sui metadati essenziali. La consultazione è veloce,

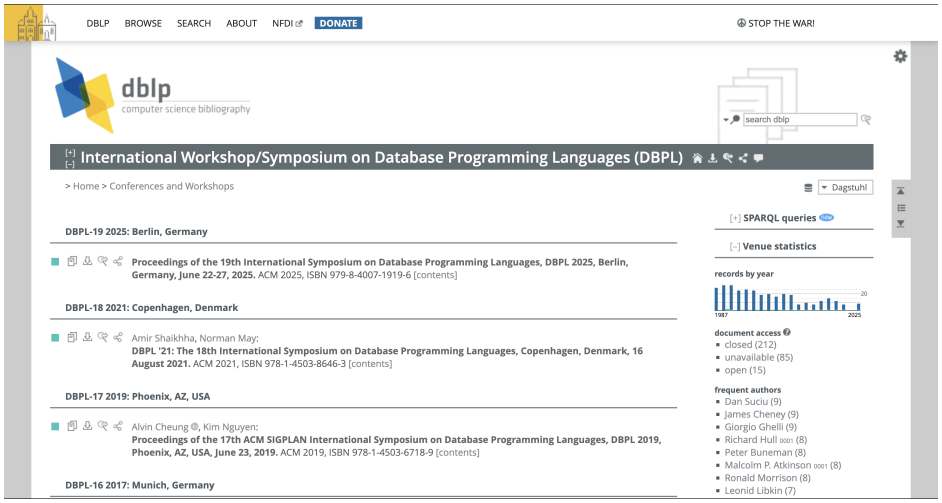

Figura 1.3: Interfaccia della schermata di ricerca su DBLP.

ma per ottenere informazioni su citazioni e qualità delle sedi è necessario integrare manualmente i dati con altri portali.

CORE e SCImago propongono interfacce specializzate: il primo fornisce tabelle di conferenze con il relativo ranking; il secondo consente di esplorare riviste e paesi in base a SJR, quartili e altri indicatori. Entrambi sono costruiti per analisi puntuali sulle venue, ma non offrono una vista centrata sull'autore o sul gruppo di ricerca.

Questa frammentazione si traduce in un'esperienza d'uso disomogenea e confusionaria: per ricostruire il profilo bibliometrico di un ricercatore occorre passare da un portale all'altro, spesso ripetendo le stesse ricerche (nome autore, ISSN, titolo della conferenza) su interfacce e modelli di navigazione differenti.

Frammentazione dei dati, delle interfacce e della semantica

La frammentazione osservata non riguarda esclusivamente la distribuzione delle informazioni tra portali differenti, ma si manifesta anche a livello di

struttura dei dati e del significato degli indicatori. In particolare, si possono distinguere tre livelli:

- **frammentazione dei dati**: ogni portale fornisce solo una porzione delle informazioni utili, con coperture e criteri diversi;
- **frammentazione dell'interazione**: l'utente deve navigare interfacce differenti, ripetendo ricerche e confronti;
- **frammentazione semantica**: gli stessi concetti possono essere rappresentati in modi differenti (venue, tipologie, identificativi), rendendo difficile l'allineamento.

Questi aspetti aumentano il rischio di interpretazioni parziali. Ad esempio, la sola consultazione di DBLP fornisce una lista accurata delle pubblicazioni in informatica, ma non permette di valutarne l'impatto citazionale; viceversa, una consultazione basata solo su Scopus può restituire metriche avanzate ma una copertura incompleta delle conferenze.

DocRanks 2.0 nasce anche per colmare questo divario a livello di interfaccia. La piattaforma si pone come "strato di integrazione" sopra i diversi portali bibliometrici, con i seguenti obiettivi:

- integrare in un'unica vista i dati estratti da Scopus, DBLP, CORE e SCImago;
- presentare per ogni autore un profilo completo che combini produttività, citazioni, metriche normalizzate e informazioni qualitative sulle sedi;
- permettere filtri, ordinamenti e confronti tra pubblicazioni senza cambiare continuamente piattaforma;
- offrire un'interfaccia coerente e uniforme, in cui il passaggio tra dashboard, vista autore, elenco pubblicazioni e Chat AI avviene tramite un modello di interazione consistente e facilmente apprendibile.

L'obiettivo non è sostituire i portali originali, ma ridurre i passaggi necessari tra strumenti diversi e fornire una sintesi operativa che migliori la consultazione e il confronto dei dati; le scelte tecnologiche e architetturali che rendono possibile tale integrazione verranno approfondite nei capitoli successivi. Verranno anche mostrate le schermate relative ad ogni pagina implementata nella nuova versione di DocRanks.

Capitolo 2

Problema, Obiettivi e Design della Soluzione

Questo capitolo formalizza il problema affrontato da DocRanks 2.0, definisce gli obiettivi del progetto e descrive il design della soluzione proposta a un livello architetturale e concettuale. Il capitolo si concentra sul *cosa* e sul *perché* (esigenze, vincoli, scelte di design), mentre il *come* (tecnologie, implementazione e struttura della codebase) viene trattato nel Capitolo 3.

2.1 Problema

La consultazione e la valutazione della produzione scientifica richiedono spesso di combinare informazioni provenienti da fonti diverse.

Nella pratica, per ottenere un quadro utilizzabile (lista pubblicazioni, metriche, qualità delle venue) l'utente è costretto a interrogare più portali separati, ripetere ricerche simili su interfacce differenti, riconciliare manualmente risultati che possono differire per copertura e rappresentazione e trascrivere o esportare dati con un rischio non trascurabile di errori e incoerenze.

Il problema non è solamente "avere tanti dati", ma **renderli consultabili in modo coerente**, mantenendo visibile l'origine delle informazioni e riducendo l'attrito operativo per l'utente. [5]

Oltre alla frammentazione dell'interazione, emergono difficoltà tipiche dell'integrazione di dati bibliografici:

- **disallineamenti tra fonti**: conteggi, liste e metadati possono differire anche per lo stesso autore;
- **eterogeneità di rappresentazione**: lo stesso concetto (venue, tipologia di pubblicazione, identificativo) può essere espresso in modi differenti;
- **ambiguità e varianti testuali**: acronimi, edizioni e differenze di naming rendono non banale l'associazione tra dati di provider diversi;
- **aggiornamenti non sincroni**: le fonti evolvono nel tempo con frequenze differenti, rendendo instabile una consultazione "al volo" basata su interrogazioni ripetute.

DocRanks 2.0 nasce per fornire una **vista unificata** orientata alla consultazione e al confronto, riducendo il numero di passaggi necessari e offrendo un'esperienza d'uso uniforme, senza sostituire le fonti originali ma agendo come livello di integrazione.

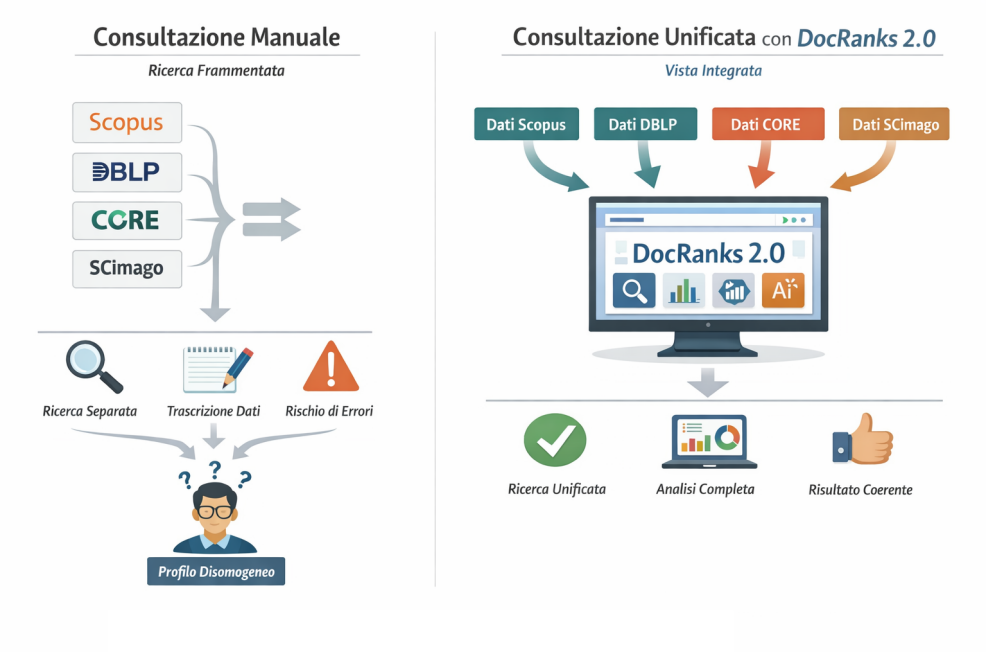

Figura 2.1: Frammentazione operativa nella consultazione manuale di più portali e vista unificata ottenibile tramite DocRanks 2.0.

2.1.1 Perché la frammentazione è un problema: costo operativo e rischio di errore

La consultazione manuale non ha solo un costo temporale: introduce un costo *cognitivo* e un rischio di errore. L'utente deve ricordare quali informazioni sono disponibili su ciascun portale, come interpretarle e come collegarle correttamente. Inoltre, anche quando i portali consentono esportazioni, l'utente resta responsabile della riconciliazione e della coerenza tra dataset.

Un ulteriore elemento riguarda la **ripetibilità** dell'analisi. Se due consultazioni vengono eseguite in momenti diversi, i risultati possono variare a causa di aggiornamenti delle fonti o differenze nei criteri di copertura, rendendo difficile ottenere un quadro stabile senza una persistenza locale. DocRanks 2.0 affronta quindi un problema di integrazione orientata all'uso:

l'obiettivo è rendere il dato *consultabile* e *verificabile* all'interno di un flusso uniforme.

2.1.2 Dimensioni del problema: dati, interazione e semantica

La frammentazione si manifesta su tre dimensioni complementari:

Frammentazione dei dati ogni fonte fornisce solo una parte delle informazioni utili, con coperture e criteri differenti.

Frammentazione dell'interazione l'utente deve navigare interfacce diverse, ripetendo ricerche e confronti.

Frammentazione semantica gli stessi concetti possono essere rappresentati in modi diversi (venue, tipologie, identificativi), rendendo difficile l'allineamento.

Queste dimensioni motivano la necessità di un livello di integrazione che non sia un semplice "aggregatore": per essere affidabile serve un modello interno coerente, regole di normalizzazione, persistenza e gestione esplicita dell'incertezza.

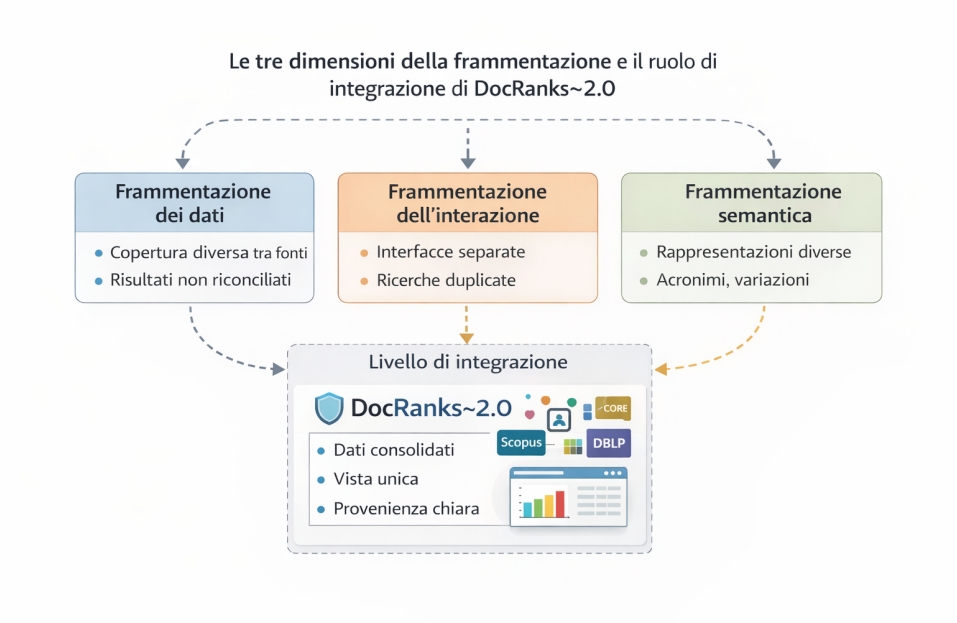

Figura 2.2: Le tre dimensioni della frammentazione (dati, interazione, semantica) e il ruolo di DocRanks 2.0 come livello di integrazione.

2.2 Obiettivi del progetto

Gli obiettivi del progetto sono stati distinti in **funzionali** e **non funzionali**, secondo una classificazione comunemente adottata nell'ingegneria del software.

Gli **obiettivi funzionali** descrivono *cosa* il sistema deve fare dal punto di vista dell'utente e dei casi d'uso: quali operazioni devono essere supportate, quali informazioni devono essere fornite e quali funzionalità devono essere rese disponibili attraverso l'interfaccia. Essi definiscono quindi il comportamento osservabile dell'applicazione e rappresentano i requisiti direttamente percepibili dall'utilizzatore finale.

Gli **obiettivi non funzionali**, invece, riguardano *come* tali funzionalità devono essere offerte. Essi includono aspetti trasversali come prestazioni, robustezza, manutenibilità, estendibilità, tracciabilità dei dati e riproducibilità

dell'ambiente. Pur non essendo immediatamente visibili all'utente, questi requisiti influenzano in modo determinante la qualità complessiva del sistema e ne condizionano l'evoluzione nel tempo.

La distinzione tra obiettivi funzionali e non funzionali consente di separare chiaramente le esigenze operative da quelle architetturali e qualitative, fornendo una base strutturata per il design della soluzione descritta nelle sezioni successive.

2.2.1 Obiettivi funzionali

1. **Importare e consolidare un profilo autore**: acquisire un insieme coerente di metadati e indicatori, mantenendo un riferimento chiaro alla fonte. [3]

2. **Esplorare le pubblicazioni**: fornire un elenco consultabile con filtri, ordinamenti e dettaglio sufficiente per l'analisi.

3. **Arricchire le pubblicazioni con informazioni sulle venue**: associare indicatori qualitativi quando disponibili, gestendo differenze di nomenclatura e ambiguità. [4]

4. **Supportare il confronto e la lettura sintetica**: rendere immediata la comprensione del profilo attraverso aggregazioni e indicatori principali.

5. **Integrare un assistente conversazionale**: permettere interrogazioni guidate e supporto alla consultazione, mantenendo persistenza e contesto. [6]

2.2.2 Obiettivi non funzionali

1. **Coerenza e tracciabilità**: ogni valore mostrato deve essere riconducibile alla fonte che lo fornisce.

2. **Robustezza verso fonti esterne**: gestire errori di rete, indisponibilità temporanee e vincoli di utilizzo senza compromettere l'esperienza complessiva. [3]

3. **Prestazioni e usabilità**: garantire navigazione fluida, caricamenti progressivi e interazioni rapide anche su liste numerose.

4. **Manutenibilità ed estendibilità**: rendere semplice aggiungere nuove fonti, indicatori o viste senza reintrodurre accoppiamenti monolitici.

5. **Riproducibilità dell'ambiente**: facilitare avvio e installazione dell'intero sistema per sviluppo e consegna. [7]

2.2.3 Requisiti di consultazione: uniformità, verificabilità e gestione dell'incertezza

Per soddisfare gli obiettivi senza introdurre ambiguità, DocRanks 2.0 adotta tre principi guida:

Uniformità l'utente deve poter consultare profilo e pubblicazioni con una navigazione e una struttura informativa coerenti, senza dover conoscere le differenze operative tra fonti.

Verificabilità le informazioni devono essere interpretabili: quando esistono più possibili valori o disallineamenti, il sistema deve evitare di "nasconderli" e deve rendere chiara la provenienza.

Incertezza esplicita in presenza di match ambigui (es. venue non riconosciuta o più associazioni plausibili), il sistema deve trattare l'incertezza come stato previsto, non come errore da mascherare.

Questi requisiti influenzano direttamente il design: persistenza locale, normalizzazione e meccanismi di logging non sono elementi accessori, ma componenti necessari per offrire una consultazione stabile. [8]

2.3 Stakeholder e casi d'uso

DocRanks 2.0 è progettato principalmente per:

- ricercatori che vogliono consultare rapidamente il proprio profilo o quello di colleghi;
- gruppi di ricerca che necessitano di una panoramica confrontabile su più autori;
- contesti accademici/didattici in cui sia utile mostrare metriche e venue in modo sintetico e navigabile.

I casi d'uso principali sono:

UC1 – Importazione autore L'utente inserisce l'identificativo autore e avvia l'importazione o l'aggiornamento. [3]

UC2 – Consultazione profilo L'utente visualizza metriche sintetiche, metadati e riepiloghi.

UC3 – Esplorazione pubblicazioni L'utente filtra/ordina la lista e analizza le venue.

UC4 – Arricchimento qualitativo Il sistema associa indicatori alle venue e li presenta in modo interpretabile. [4]

UC5 – Chat di supporto L'utente pone domande e ottiene risposte contestualizzate ai dati disponibili. [6]

Questi casi d'uso sono progettati per un utilizzo iterativo: l'utente alterna consultazione sintetica e dettaglio, ripetendo importazioni/aggiornamenti senza perdere coerenza.

2.3.1 Scenari d'uso tipici

Scenario A – consultazione rapida un ricercatore vuole controllare rapidamente profilo e pubblicazioni senza ripetere ricerche su portali diversi.

Scenario B – confronto tra autori un gruppo vuole confrontare più profili con indicatori qualitativi visibili e coerenti.

Scenario C – lettura guidata un utente vuole ottenere una sintesi (es. trend recenti, distribuzione per venue) senza ricostruire manualmente il dato.

Figura 2.3: Esempio del flusso di una tipica di user journey.

2.4 Vincoli e assunzioni

Questa sezione definisce vincoli e assunzioni a livello di **dominio e problema**, indipendenti da scelte implementative specifiche. Tali elementi deri-

vano dalla natura delle fonti bibliometriche e dal fatto che DocRanks 2.0 opera come livello di integrazione sopra sistemi esterni, ciascuno caratterizzato da coperture, formati e convenzioni differenti.

DocRanks 2.0 si confronta innanzitutto con l'**eterogeneità dei formati**: informazioni concettualmente simili possono essere rappresentate in modi differenti, con campi opzionali, granularità diverse o strutture non uniformi. A questo si aggiunge una **copertura non uniforme** delle fonti, per cui liste di pubblicazioni, conteggi e classificazioni possono variare anche a parità di autore o periodo considerato. Ulteriori vincoli sono imposti direttamente dai provider esterni, che introducono **limitazioni di utilizzo** legate ad autenticazione, frequenza delle richieste e condizioni di accesso [3]. Un aspetto particolarmente critico riguarda inoltre la **ambiguità sulle venue**: acronimi, edizioni annuali e variazioni testuali rendono non banale l'allineamento tra record che rappresentano la stessa sede editoriale. Infine, non sempre sono disponibili **identificativi stabili** e univoci che consentano di risolvere i match in modo diretto, rendendo necessario trattare esplicitamente casi di incertezza.

Alla luce di questi vincoli, la soluzione proposta assume la necessità di un livello di **consolidamento locale** dei risultati, così da rendere la consultazione ripetibile nel tempo e mitigare l'instabilità dovuta ad aggiornamenti asincroni e vincoli di accesso alle fonti [8].

2.4.1 Implicazioni progettuali dei vincoli

I vincoli descritti guidano scelte progettuali precise a livello concettuale. Se una fonte esterna può essere temporaneamente indisponibile o soggetta a limitazioni, la consultazione deve potersi appoggiare su dati già consolidati, mantenendo un comportamento prevedibile per l'utente. Analogamente, in presenza di venue ambigue, il sistema deve distinguere tra associazioni certe e associazioni incerte, evitando inferenze non verificabili che potrebbero compromettere l'interpretazione dei risultati. Infine, quando le fonti differiscono per copertura o criteri di indicizzazione, la presentazione deve rendere espli-

cita la provenienza delle metriche, così da consentire all'utente di interpretare correttamente eventuali disallineamenti.

2.5 Panoramica della soluzione

DocRanks 2.0 è progettato come sistema a componenti, con una separazione chiara tra interfaccia utente, servizi applicativi e persistenza. L'interfaccia utente è orientata alla consultazione, al filtraggio e alla navigazione dei dati, includendo una funzionalità di chat come supporto all'interpretazione [1, 9, 10]. I servizi applicativi orchestrano i flussi principali del sistema, occupandosi dell'integrazione delle fonti esterne, della normalizzazione e della composizione delle informazioni. La persistenza mantiene i dati consolidati, gli arricchimenti, le sessioni e i log, fungendo da base stabile per la consultazione [8].

A livello concettuale, il sistema acquisisce dati esterni e li trasforma in un **modello interno coerente**; persiste localmente i risultati per ridurre latenza e dipendenza dalle fonti; espone le funzionalità alla UI tramite un **contratto di servizi** stabile [2]; infine presenta una vista uniforme in cui indicatori quantitativi e qualitativi sono affiancati e interpretabili.

Il principio guida è che l'esperienza di consultazione debba poggiare su un dato *stabile e verificabile*, demandando alle interrogazioni verso i provider esterni il ruolo di aggiornamento e integrazione, e non quello di supporto continuo alla navigazione.

2.6 Aspetti di design della UI

In un sistema orientato alla consultazione e al confronto, il design dell'interfaccia incide direttamente sul costo cognitivo e sulla capacità dell'utente di interpretare correttamente metriche, liste e indicatori qualitativi. Poiché DocRanks 2.0 integra fonti diverse e potenzialmente disallineate, la UI deve ridurre l'attrito operativo e rendere evidenti sia la provenienza sia la natura

del dato visualizzato, in coerenza con i principi di uniformità e verificabilità introdotti nelle sezioni precedenti.

Un primo criterio di progettazione riguarda la **progressività dell'informazione**: l'utente deve poter partire da una vista sintetica del profilo autore e accedere gradualmente a livelli di dettaglio maggiori, come le singole pubblicazioni, le venue e gli indicatori associati, senza perdere il contesto di navigazione. Ciò implica che filtri, ordinamenti e paginazione siano elementi strutturali dell'interazione e non funzionalità accessorie.

Un secondo criterio riguarda la **leggibilità della provenienza**. Poiché indicatori quantitativi e qualitativi possono derivare da provider differenti, la UI deve consentire di distinguere chiaramente l'origine dei valori mostrati, evitando di combinare implicitamente informazioni eterogenee e riducendo il rischio di interpretazioni fuorvianti in presenza di disallineamenti.

Infine, l'introduzione di una chat integrata richiede coerenza con le viste tradizionali di consultazione. La chat deve agire come supporto alla lettura e alla sintesi dei dati, e non come interfaccia alternativa isolata. In particolare, l'utente deve poter alternare navigazione e interrogazione guidata mantenendo lo stesso contesto informativo (profilo visualizzato, filtri applicati, lista corrente), così da utilizzare la conversazione come strumento di interpretazione, e non come canale separato.

Questi criteri sono coerenti con un approccio SPA, in cui l'interazione avviene tramite caricamenti selettivi e aggiornamenti progressivi dell'interfaccia, preservando continuità e fluidità nella navigazione [1, 11].

2.7 Architettura logica

2.7.1 Componenti principali

Coerentemente con la panoramica introdotta nella sezione precedente, l'architettura logica di DocRanks 2.0 può essere letta come una composizione di componenti che collaborano secondo responsabilità distinte, così da

mantenere separati i livelli di presentazione, orchestrazione e gestione del dato.

Dal punto di vista dell'utente, la **UI** costituisce lo strato di consultazione e interazione: organizza la navigazione tra profilo, pubblicazioni e dettagli di venue, e integra la chat come supporto alla lettura dei risultati [11]. La UI non ha il compito di "costruire" il dato, ma di presentarlo in modo progressivo e coerente, rendendo chiari filtri applicati e contesto di consultazione.

La logica applicativa risiede nel livello di **API e servizi**, che espone verso il frontend un contratto stabile e incapsula la composizione delle informazioni necessarie alle viste. Questo livello orchestra i flussi principali (ad esempio importazione, consultazione e arricchimento) e centralizza validazione e gestione degli errori, mantenendo un punto unico di responsabilità rispetto alle risorse esposte [2].

Le **integrazioni con le fonti esterne** rappresentano il canale di acquisizione dei dati bibliografici e bibliometrici: gestiscono il recupero, il parsing delle risposte e l'adattamento ai vincoli imposti dai provider, incluse eventuali indisponibilità temporanee o condizioni di utilizzo [3, 4]. La loro separazione dal livello di servizio consente di controllare l'impatto di cambiamenti esterni, preservando la stabilità del contratto verso la UI.

Un ruolo trasversale è svolto dalla componente di **normalizzazione e matching**, che rende confrontabili rappresentazioni eterogenee dello stesso concetto (in particolare per venue e metadati) e abilita l'arricchimento qualitativo in modo coerente. In altre parole, questa componente evita che differenze testuali o strutturali tra fonti si traducano in incoerenze nella consultazione.

Infine, la **persistenza** costituisce l'archivio locale su cui si basa la consultazione ordinaria: conserva dati consolidati, arricchimenti, stato delle importazioni e tracciamento operativo, oltre alle conversazioni della chat. Questo consente di ridurre la dipendenza runtime dalle fonti esterne e di rendere le operazioni ripetibili e verificabili [8].

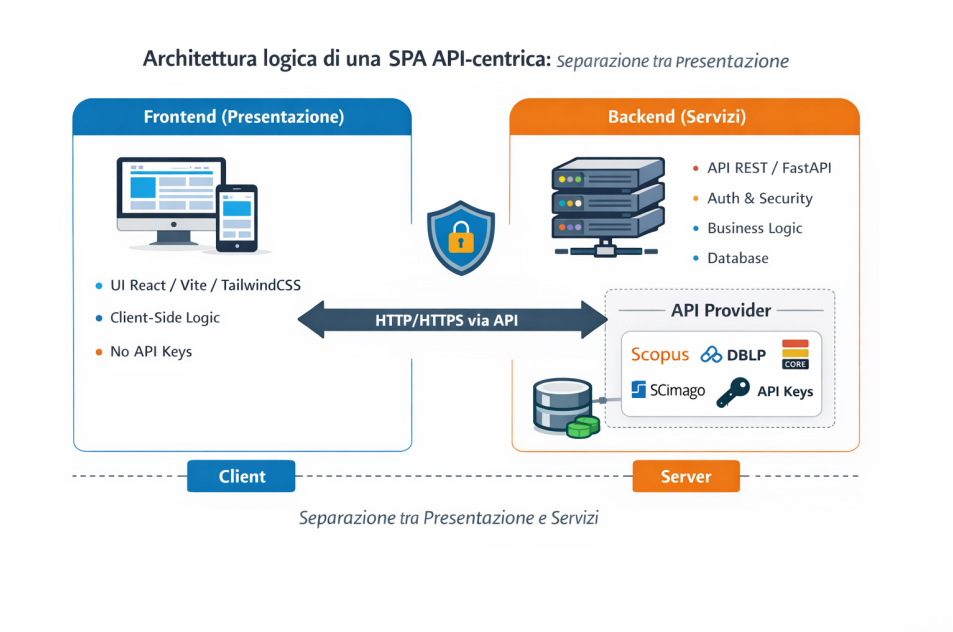

Figura 2.4: Architettura logica: separazione tra UI, servizi applicativi e persistenza, con integrazione di fonti esterne.

2.7.2 Principi di responsabilità e riduzione dell'accoppiamento

Il design architetturale di DocRanks 2.0 è guidato dal principio di separazione delle responsabilità, con l'obiettivo di ridurre l'accoppiamento tra le diverse parti del sistema e facilitare l'evoluzione indipendente dei singoli componenti.

In particolare, l'interfaccia utente è concepita come uno strato focalizzato esclusivamente su presentazione e interazione: essa gestisce la navigazione, l'applicazione dei filtri e la visualizzazione dei risultati, senza incorporare logiche di composizione o di integrazione dei dati. Tale scelta consente di mantenere la UI semplice, prevedibile e facilmente modificabile al variare delle esigenze di consultazione.

Le regole di coerenza, composizione e normalizzazione del dato sono invece centralizzate nei servizi applicativi. Questo livello rappresenta il punto in

cui le informazioni provenienti da fonti diverse vengono validate, allineate e rese coerenti con il modello interno, evitando la duplicazione di logica e garantendo un comportamento uniforme verso tutte le viste.

L'accesso alle fonti esterne è infine isolato in moduli dedicati, così da contenere l'impatto di errori, cambiamenti nelle API o limitazioni di utilizzo. In questo modo, eventuali variazioni a livello di integrazione non propagano effetti indesiderati verso l'interfaccia o la persistenza, preservando la stabilità complessiva del sistema.

Questa organizzazione favorisce un'evoluzione controllata dell'architettura e riduce il rischio di interventi "a cascata", rendendo il sistema più manutenibile e robusto nel tempo.

2.8 Flussi principali

2.8.1 Flusso di importazione e aggiornamento autore (UC1)

Il flusso di importazione e aggiornamento è progettato per essere ripetibile, robusto e osservabile. A seguito dell'inserimento di un identificativo autore, il sistema verifica la presenza di dati già consolidati e determina se procedere con una nuova importazione o con un aggiornamento incrementale.

I moduli di integrazione recuperano quindi i dati dalle fonti esterne [3, 4]; tali dati vengono validati e trasformati nel modello interno, applicando le regole di normalizzazione e composizione definite nei servizi applicativi. Una volta completata l'elaborazione, i risultati vengono persistiti localmente e l'esito dell'operazione (successo o errore) viene registrato con informazioni utili alla diagnosi [8]. Durante l'intero processo, la UI fornisce un feedback interpretabile sullo stato dell'operazione.

La presenza di una persistenza locale consente di ridurre richieste duplicate durante la consultazione, mantenere un comportamento stabile anche in

presenza di vincoli o disservizi esterni [3] e rendere ripetibili le operazioni di aggiornamento senza generare duplicazioni o stati incoerenti.

2.8.2 Flusso di consultazione profilo e pubblicazioni (UC2–UC3)

La consultazione è progettata come un flusso di lavoro continuo: l'utente parte dal profilo autore, applica filtri e ordinamenti alla lista pubblicazioni e accede ai dettagli delle venue mantenendo invariato il contesto della sessione di navigazione.

Il passaggio tra riepilogo e dettaglio avviene in modo fluido, e gli indicatori qualitativi associati alle venue sono presentati come supporto interpretativo, non come sostituto del giudizio dell'utente. L'obiettivo è consentire una lettura comparativa e consapevole dei risultati, senza richiedere la conoscenza delle differenze operative tra le fonti di origine.

2.8.3 Arricchimento qualitativo e normalizzazione (UC4)

L'arricchimento qualitativo delle pubblicazioni richiede un processo di normalizzazione e matching, reso necessario dalle differenze di rappresentazione delle venue nelle varie fonti. Le difficoltà principali derivano da variazioni testuali, acronimi, denominazioni storiche o dalla mancanza di identificativi stabili.

Per affrontare questo problema, il sistema adotta una strategia a livelli. In primo luogo, viene applicata una normalizzazione deterministica delle stringhe; quando disponibili, vengono utilizzati identificativi affidabili; le associazioni risolte vengono poi memorizzate per evitare ripetizioni. Nei casi in cui non sia possibile individuare un'associazione univoca, il sistema mantiene l'esito come *ambiguità* o *assenza di match*, evitando inferenze non verificabili.

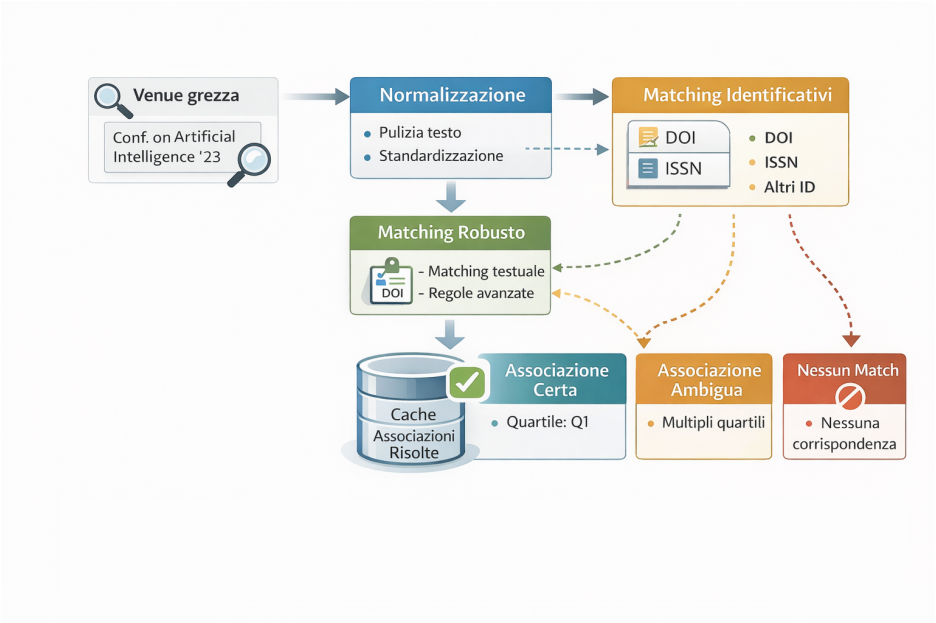

Figura 2.5: Strategia a livelli per normalizzazione e matching delle venue: dal dato grezzo all'associazione certa, ambigua o assente, con memorizzazione delle associazioni risolte.

2.8.4 Chat di supporto e persistenza (UC5)

La chat è progettata come funzionalità complementare alla consultazione tradizionale. L'utente può creare o selezionare una sessione e interagire con il sistema per ottenere chiarimenti, sintesi o letture guidate dei dati disponibili.

Ogni messaggio viene persistito con riferimento alla sessione, consentendo continuità tra accessi e una gestione controllata del contesto conversazionale. La persistenza della chat è necessaria non solo per l'esperienza d'uso, ma anche per supportare audit, debugging e interpretazione delle risposte generate [6, 8].

2.9 Modello concettuale dei dati

Il modello concettuale di DocRanks 2.0 è organizzato attorno a un numero limitato di entità fondamentali, progettate per supportare consultazione, tracciabilità e arricchimento del dato.

Le entità principali includono l'autore, con i relativi identificativi e metriche aggregate; le pubblicazioni, descritte tramite metadati e collegate a venue e autori; le venue stesse, arricchite da indicatori qualitativi derivati da fonti esterne; e le strutture di supporto per il tracciamento delle importazioni, dei log operativi e delle conversazioni della chat.

La progettazione del modello segue tre principi guida: i dati devono essere interrogabili tramite filtri e ordinamenti, coerenti rispetto alle informazioni consolidate e, coerentemente con i requisiti di verificabilità discussi in precedenza, mantenere un riferimento esplicito alla fonte di provenienza.

2.10 Strategie trasversali

2.10.1 Tracciabilità della fonte

Quando valori simili esistono su più fonti, DocRanks 2.0 evita di "mescolare" metriche in modo implicito. Ogni valore mostrato deve essere ricon-

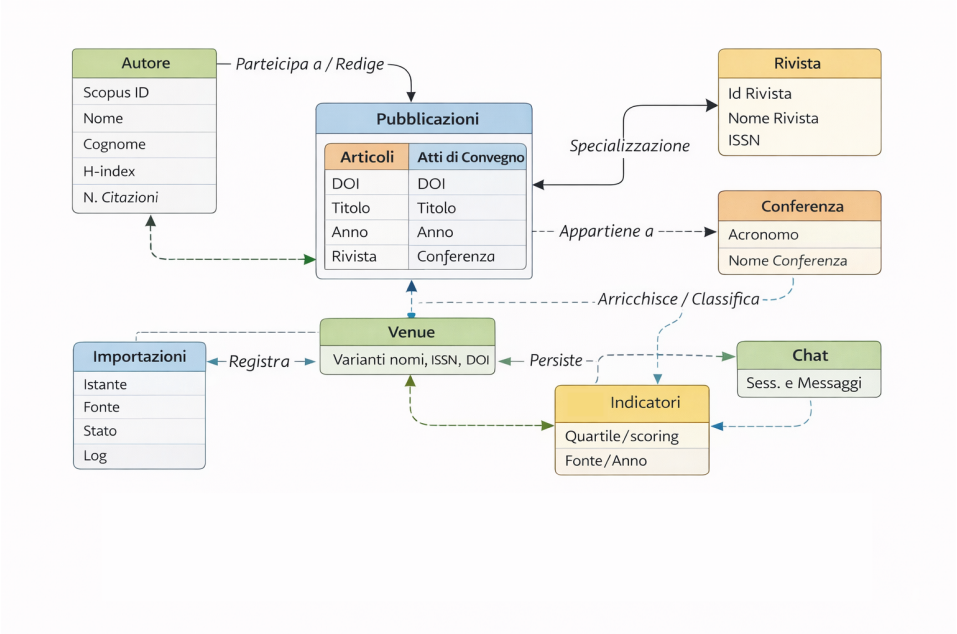

Figura 2.6: Modello concettuale semplificato: entità principali e relazioni (autore, pubblicazioni, venue, indicatori, importazioni, chat).

ducibile alla sorgente, così da rendere interpretabili differenze tra conteggi e evitare confronti fuorvianti.

2.10.2 Riduzione della dipendenza runtime dalle fonti

Coerentemente con i vincoli di dominio discussi in precedenza, l'architettura utilizza la persistenza locale come base primaria per la consultazione, riservando le chiamate alle fonti esterne alle fasi di importazione e aggiornamento [3, 8].

2.10.3 Gestione errori e robustezza

Il sistema deve degradare in modo controllato:

- errori di integrazione non devono corrompere lo stato locale;
- l'utente deve ricevere messaggi chiari e azionabili;

- le operazioni devono essere ripetibili senza generare duplicazioni o stati incoerenti.

2.10.4 Osservabilità e tracciabilità delle operazioni

Operazioni come importazioni e aggiornamenti devono essere osservabili per supportare debugging e interpretazione dei risultati. La possibilità di ricostruire lo stato di una importazione (esito, fonte coinvolta, data, motivazione del fallimento) migliora l'affidabilità percepita e rende il sistema utilizzabile anche in condizioni non ideali.

2.11 Sintesi e transizione al Capitolo 3

In questo capitolo sono stati definiti problema, obiettivi e design della soluzione DocRanks 2.0, evidenziando flussi e strategie per integrare fonti eterogenee mantenendo coerenza e tracciabilità.

Il Capitolo 3 descrive come questo design è stato realizzato: tecnologie adottate, struttura della codebase, implementazione dei moduli principali e scelte operative più rilevanti.

Capitolo 3

Progetto DocRanks 2.0

In questo capitolo viene descritto come le scelte concettuali e architetturali introdotte nel Capitolo 2 sono state concretamente realizzate nella nuova versione di DocRanks 2.0, con particolare attenzione all'organizzazione del codice, ai principali moduli implementati e alle soluzioni adottate per gestire l'integrazione con le fonti bibliometriche e la *Chat AI*.

L'obiettivo non è ripetere le motivazioni già discusse in precedenza, ma illustrare in modo sistematico il percorso di migrazione dalla precedente applicazione PHP a un sistema moderno basato su backend Python e *Single Page Application* (SPA), evidenziando le principali decisioni progettuali e le implicazioni pratiche per manutenibilità, estendibilità e qualità del codice.

3.1 Obiettivi della nuova architettura

La riprogettazione di DocRanks è stata guidata dall'esigenza di superare i limiti strutturali della versione originaria basata su PHP server-side e di allineare l'applicazione ai requisiti emersi nel processo di analisi del problema.[5]

Oltre agli obiettivi generali già introdotti, la progettazione ha richiesto di definire in modo più fine quali responsabilità attribuire a ciascun componente del sistema. In particolare, è stato necessario chiarire quali trasformazioni

dovessero essere eseguite lato backend, quali aspetti fossero delegati al database e quali logiche dovessero rimanere esclusivamente nel frontend, evitando sovrapposizioni e duplicazioni.[8, 12]

Gli obiettivi principali possono essere riassunti in quattro direttrici operative:

- **separazione delle responsabilità**, tramite una distinzione chiara tra interfaccia utente, logica applicativa e persistenza dei dati;
- **modularità ed estendibilità**, con componenti indipendenti in grado di evolvere senza introdurre un nuovo monolite;
- **integrazione robusta delle fonti bibliometriche**, gestendo in modo esplicito limiti di utilizzo, formati eterogenei e disallineamenti semantici;[3, 4]
- **riproducibilità del deployment**, tramite containerizzazione dell'intero stack applicativo.[7]

Questi obiettivi hanno portato all'adozione di un backend API-centrico in Python, di una SPA sviluppata con React e di un database relazionale MySQL come base di persistenza locale per dati, arricchimenti e sessioni di chat.[8, 11, 12]

Per rendere più chiaro il cambiamento introdotto, la Figura 3.1 propone un confronto concettuale tra la vecchia architettura monolitica e quella nuova basata su servizi e SPA.

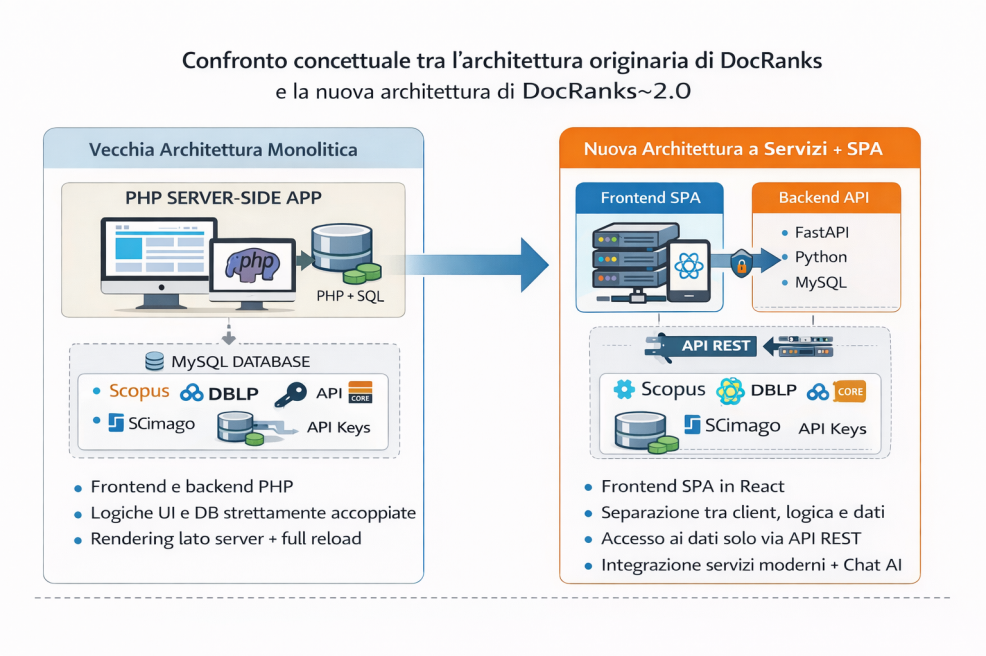

Figura 3.1: Confronto concettuale tra l'architettura originaria di DocRanks e la nuova architettura di DocRanks 2.0.

3.2 Analisi della versione originaria

Prima di procedere con la migrazione è stata condotta un'analisi sistematica della versione originaria di DocRanks, sviluppata come applicazione PHP tradizionale con rendering server-side e accesso diretto al database MySQL.[8]

Struttura della codebase e accoppiamento

La codebase precedente seguiva un modello page-oriented: ogni pagina PHP era responsabile dell'intero flusso *input* → *interrogazione SQL* → *elaborazione* → *output HTML*.[13]

Questo modello comportava alcune conseguenze pratiche:

- logica applicativa, accesso ai dati e presentazione erano mescolati all'interno degli stessi file;

- porzioni di codice simili venivano duplicate in pagine diverse, rendendo complessa l'evoluzione funzionale;
- l'assenza di un livello di servizio centrale rendeva difficile introdurre regole uniformi di validazione, normalizzazione e gestione degli errori.

Dal punto di vista manutentivo, modifiche anche localizzate potevano richiedere interventi su più pagine, aumentando il rischio di regressioni e incoerenze. Inoltre, la mancanza di una struttura di progetto chiara rendeva poco agevole l'onboarding di nuovi sviluppatori, che dovevano necessariamente ricostruire a posteriori le convenzioni implicite della codebase.

Domino implicito e flussi sincroni

Concetti centrali come autore, pubblicazione, venue e importazioni non erano modellati come entità di dominio esplicite, ma emergevano implicitamente dalle query SQL e dai frammenti di codice presenti nelle pagine.[14]

Questo approccio aveva due effetti principali:

- la logica di trasformazione dei dati era ripetuta in più punti con varianti locali;
- risultava complesso introdurre test automatici, poiché la logica applicativa era fortemente dipendente dal contesto di rendering.

Inoltre, ogni interazione dell'utente comportava il ricaricamento completo della pagina, un modello poco adatto a funzionalità che richiedono osservabilità e progressione come l'importazione di profili autore o l'aggiornamento massivo delle pubblicazioni.

Per rendere evidente questo aspetto, il Listato 3.1 riporta uno pseudocodice rappresentativo della versione legacy, in cui una singola richiesta HTTP esegue logica applicativa, chiamate a servizi esterni e generazione dell'HTML nella stessa interazione. In questo modello, ogni azione dell'utente comporta il ricaricamento completo della pagina e l'assenza di un livello intermedio dedicato alla gestione dello stato o della progressione dell'operazione.

Listing 3.1: Pseudocodice semplificato della versione PHP legacy con logica e rendering accoppiati.

```
<?php
// INPUT dalla richiesta HTTP
authorId = $_GET['author_id']

// LOGICA + I/O (es. chiamata API o query DB)
results = externalServiceCall(authorId)
filtered = filterResults(results)

// RENDERING HTML nella stessa richiesta
foreach (filtered as row) {
    print "<tr><td>" + row.title + "</td></tr>"
}
?>
```

Questa analisi ha confermato che il limite principale non risiedeva nella tecnologia in sé, ma nell'assenza di confini architetturali chiari e nella sovrapposizione tra presentazione, logica applicativa e accesso ai dati. Tale configurazione rendeva complessa l'estensione del sistema e poco adatto alla gestione di operazioni progressive o asincrone, motivando il passaggio a un backend dedicato e a una SPA client-side.[9, 10]

3.3 Struttura del repository e organizzazione dei moduli

Il progetto è organizzato in directory distinte per isolare responsabilità e semplificare manutenzione ed estensioni future. Alla radice sono presenti componenti dedicati al backend, alla pipeline ETL, alla persistenza e all'applicazione frontend.

In particolare:

- `etl/`: contiene i componenti di importazione e normalizzazione dei dati (SCImago, CORE, Scopus), insieme ai *repository* per l'accesso al database e alle logiche di *processing* degli autori (integrazione, deduplicazione, creazione delle relazioni autore–pubblicazione).
- `db/`: raccoglie asset relativi al database e script di supporto.
- `api/`: ospita moduli di esposizione API quando presenti come sottopacchetti o separazione logica.
- `frontend/`: contiene la SPA (Vite + React + TypeScript). In `frontend/src/components/` sono presenti le viste principali (es. `Dashboard.tsx`, `Settings.tsx`, `RepositoryManager.tsx`), mentre `routes/`, `contexts/`, `hooks/` e `stores/` gestiscono routing, stato applicativo e logica riutilizzabile.
- `uploads/`: directory operativa per i file utilizzati dagli import (ad esempio dataset SCImago/CORE/Scopus).
- `sql/`: script e risorse collegate allo schema del database e alla sua inizializzazione.

Completano il repository i file di configurazione per build e deployment (es. `docker-compose.yaml`, `Dockerfile`, configurazioni Vite/TypeScript/-Tailwind), che rendono riproducibile l'ambiente su macchine diverse e riducono la configurazione manuale.[7]

3.4 Backend Python e API REST

Il backend di DocRanks 2.0 è stato implementato in Python utilizzando il framework FastAPI, che fornisce tipizzazione, validazione automatica degli input e generazione della documentazione OpenAPI.[12, 15]

L'esecuzione avviene su server compatibili con lo standard ASGI, come Uvicorn, rendendo possibile la gestione efficiente di richieste concorrenti e di operazioni I/O intensive verso fonti esterne e database.[16, 17]

3.4.1 Panoramica architetturale del backend

Il backend espone una serie di endpoint REST che coprono le principali operazioni applicative: importazione dei profili autore, consultazione dei dati consolidati, arricchimento qualitativo delle venue e gestione delle sessioni di chat.[3, 12]

3.4.2 Organizzazione interna: routers, services e repositories

Per mantenere il backend evolvibile, è stata adottata una separazione esplicita in tre layer principali:[14]

routers definiscono gli endpoint HTTP, mappano le richieste verso i servizi applicativi e strutturano le risposte nel formato atteso dalla SPA;

services contengono la logica applicativa, orchestrano i flussi tra più repository e integrazioni esterne e applicano le regole di normalizzazione e composizione;

repositories incapsulano l'accesso a MySQL tramite query SQL, mantenendo il controllo esplicito sulle operazioni di lettura e scrittura.[8]

Questa organizzazione evita che la logica applicativa si concentri nei router e consente di testare in modo indipendente i servizi e i repository. A titolo esemplificativo, il Listato 3.2 mostra la definizione di un endpoint per la consultazione del profilo autore.

Listing 3.2: Esempio semplificato di endpoint FastAPI per la consultazione del profilo autore.

```
@router.get("/authors/{author_id}", response_model=
    AuthorProfileDTO)
async def get_author_profile(
    author_id: str,
    service: AuthorService = Depends(),
```

```
):
    profile = await service.get_profile(author_id)
    return profile
```

3.4.3 Endpoint principali del backend

Per documentare i flussi più rappresentativi, vengono descritti tre endpoint che coprono: (i) sintesi aggregata per dashboard, (ii) importazione e *processing* completo di un autore, (iii) interrogazione conversazionale del database tramite assistente AI.

Riepilogo per dashboard (`/api/dashboard/summary`)

L'endpoint `GET /api/dashboard/summary` restituisce una sintesi compatta del contenuto del database: numero di autori, articoli, atti di convegno, altri documenti, riviste e conferenze. Questo consente alla SPA di popolare la dashboard con poche informazioni ad alto valore informativo senza dover invocare più risorse separate.

Aggiunta e processing completo di un autore (`/api/author/add-and-process`)

L'endpoint `POST /api/author/add-and-process` gestisce un flusso end-to-end: a partire dallo `scopus_id`, il backend valida l'input, tenta di recuperare nome/cognome e metriche da Scopus, aggiorna la tabella `AUTORI` e avvia il *processing* completo tramite `AuthorProcessor`. Durante il processing vengono importate pubblicazioni e relazioni (autori–pubblicazioni), con upsert idempotenti per evitare duplicazioni. Al termine, il sistema ricalcola i conteggi per tipologia (articoli, atti, altri) e aggiorna i valori aggregati dell'autore.

Chat AI con accesso controllato al database (`/api/mcp/chat`)

L'endpoint `POST /api/mcp/chat` implementa una chat progettata per interrogare il database in modo guidato. Il backend invia al modello un prompt di sistema con schema e regole operative, abilitando il *function calling* tramite un set di tool, tra cui una funzione dedicata all'esecuzione di query SQL (`execute_sql_query`) e shortcut per statistiche ricorrenti.

Quando il modello richiede una tool call, la query viene eseguita in modalità sicura (solo `SELECT`, con blocco di keyword non ammesse e `LIMIT` automatico) e i risultati vengono ritornati al modello come JSON, che produce la risposta finale in linguaggio naturale. Messaggi e sessioni vengono persistiti nelle tabelle `CHAT_SESSIONS` e `CHAT_MESSAGES` per supportare cronologia e ripresa delle conversazioni.[6, 8]

3.4.4 Modello dei dati e persistenza

La persistenza locale rappresenta un elemento centrale di DocRanks 2.0: il database MySQL funge da *single source of truth* interna su cui si basano tutte le interazioni della SPA.[8]

Lo schema logico è stato progettato seguendo alcuni principi:

- separazione tra entità bibliografiche (`Author`, `Publication`, `Venue`) e dati derivati (`VenueMetrics`, importazioni, chat);
- tracciabilità della fonte per ogni informazione importata, per evitare di mescolare implicitamente metriche provenienti da provider diversi;[3, 4]
- supporto ad importazioni incrementali e ad aggiornamenti ripetuti senza duplicazioni.

Le entità principali includono:

- `Author`: identificativi, metadati e metriche aggregate;
- `Publication`: titoli, anni, tipo, associazioni con autore e venue;

- `Venue`: riviste e conferenze normalizzate, con riferimento alla fonte di origine;
- `VenueMetrics`: quartili SCImago, SJR e ranking CORE;[4]
- tabelle di supporto per importazioni, log operativi e conversazioni della chat.

La Figura 3.2 mostra uno schema concettuale semplificato delle entità bibliografiche e delle relazioni principali.

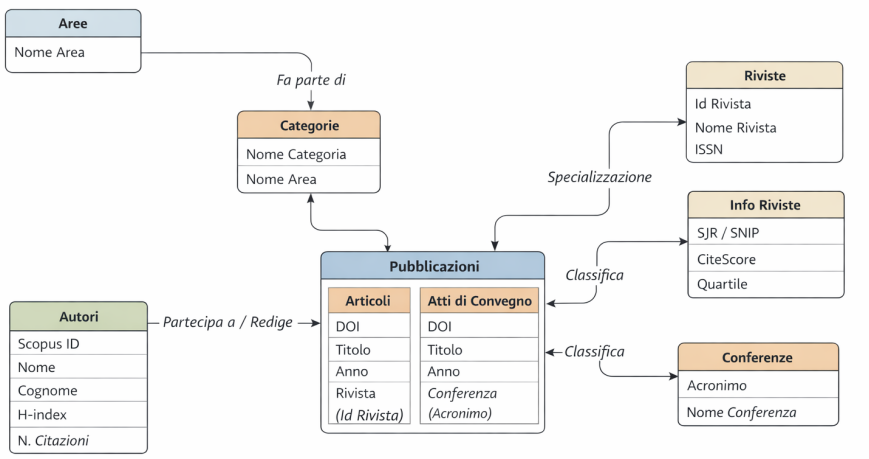

Figura 3.2: Schema concettuale semplificato delle entità bibliografiche e delle relazioni principali.

3.4.5 Integrazione delle fonti bibliometriche

Il backend funge da livello di mediazione tra la SPA e le fonti esterne, centralizzando l'accesso, la normalizzazione e la gestione degli errori.[3, 4]

Scopus. Scopus rappresenta la fonte principale per metadati di autore e pubblicazioni, nonché per le metriche quantitative quali citazioni, H-index, FWCI, CiteScore e SNIP.[3]

L'integrazione utilizza le API ufficiali e prevede:

- gestione centralizzata dell'autenticazione e dei rate limit;
- parsing e validazione delle risposte;
- persistenza dei risultati nel database locale con importazioni idempotenti.

DBLP. DBLP è utilizzato come fonte complementare per la copertura delle conferenze informatiche, tramite endpoint HTTP e dataset pubblici.[4]

I principali problemi affrontati riguardano:

- variazioni di naming delle venue e differenze tra edizioni annuali;
- assenza di identificativi stabili per alcune sedi;
- necessità di caching delle associazioni già risolte.

SCImago e CORE. SCImago Journal & Country Rank fornisce indicatori qualitativi per le riviste (SJR, quartili), mentre CORE classifica le conferenze in classi qualitative.[4]

L'assenza di API ufficiali complete ha reso necessario:

- utilizzare dataset esportati e moduli ETL dedicati;
- normalizzare le venue importate rispetto alle entità locali;
- dare priorità a rappresentazioni deterministiche per ridurre le ambiguità.

La pipeline di normalizzazione e matching delle venue è schematizzata in Figura 3.3.

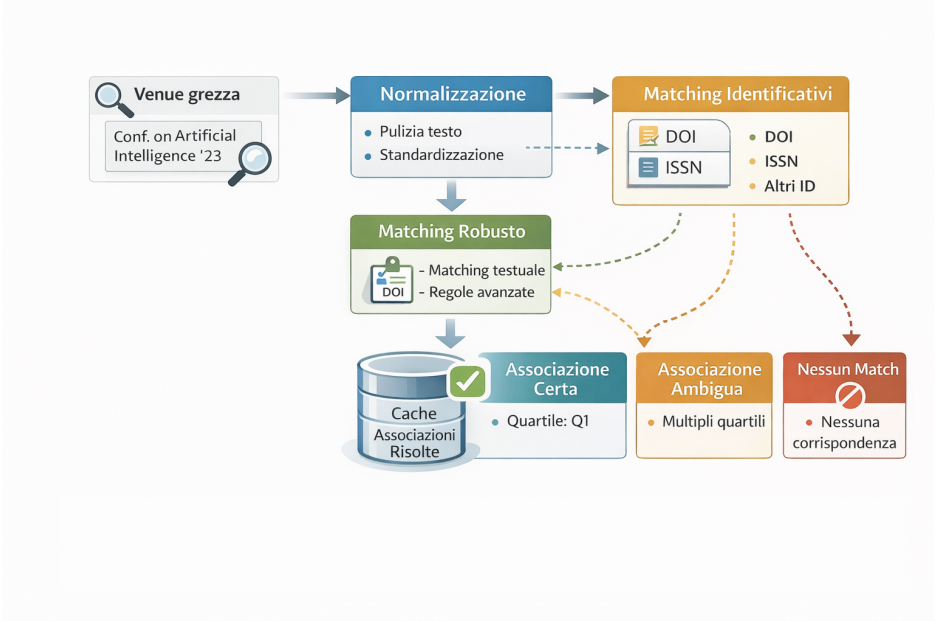

Figura 3.3: Pipeline di normalizzazione e matching delle venue tra fonti eterogenee.

3.4.6 Pipeline ETL, import e bootstrap automatico

L'acquisizione e l'aggiornamento dei dati avvengono tramite una pipeline ETL che integra più sorgenti: SCImago per metriche delle riviste, CORE per ranking delle conferenze e Scopus per metadati e metriche autore/pubblicazioni.[3]

Il backend espone endpoint di import specifici per ciascuna sorgente e un endpoint generico per upload CSV, così da supportare sia flussi automatizzati sia caricamenti manuali. Ogni import registra un evento nella tabella `ATTIVITA_LOG` con sorgente e numero di record processati, rendendo disponibile uno storico consultabile dalla dashboard.

Per facilitare il primo avvio, è presente un meccanismo di *bootstrap import*: all'avvio dell'applicazione un thread verifica se il database è vuoto e, in tal caso, esegue gli import iniziali una sola volta. Per evitare condizioni di gara in ambienti containerizzati, il sistema utilizza un lock lato database e un marker di completamento registrato nei log, così da non ripetere la procedura nei successivi restart.

3.4.7 Flussi principali lato backend

I flussi implementativi principali sono quattro e corrispondono direttamente alle esigenze individuate nella fase di analisi.

Importazione e aggiornamento autore. Dopo l'inserimento di un identificativo autore, il backend verifica la presenza di dati consolidati e decide se effettuare una nuova importazione o un aggiornamento incrementale, orchestrando le chiamate a Scopus e DBLP e persistendo l'esito dell'operazione.[3, 4, 8]

Consultazione del profilo e delle pubblicazioni. La consultazione sfrutta esclusivamente i dati presenti in MySQL: la SPA richiede al backend profili e liste paginati, con filtri e ordinamenti, evitando interrogazioni dirette verso le fonti esterne durante la navigazione.[8]

Arricchimento qualitativo. Il backend applica una pipeline di normalizzazione e matching per associare ad ogni venue gli indicatori qualitativi importati da SCImago e CORE, registrando esplicitamente i casi di ambiguità o assenza di match.[4]

Gestione della chat. Gli endpoint dedicati alla chat orchestrano la costruzione del contesto, l'invocazione dell'API OpenAI e la persistenza delle conversazioni, come descritto nella Sezione 3.6.[6, 8]

Per rendere più leggibile l'insieme di questi flussi, la Figura 3.4 mostra un diagramma che collega le principali operazioni del backend ai componenti coinvolti.

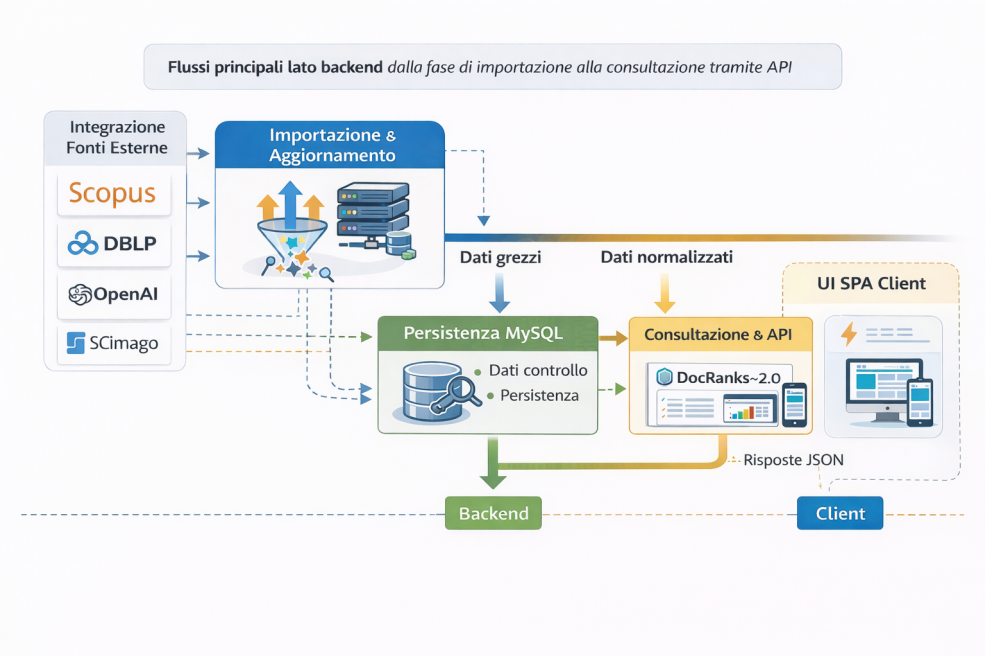

Figura 3.4: Flussi principali lato backend dalla fase di importazione alla consultazione tramite API.

3.5 Frontend SPA con React

Il frontend di DocRanks 2.0 è stato realizzato come *Single Page Application* utilizzando React, con Vite come tool di build e TailwindCSS per la definizione dello stile.[11, 18, 19]

Questa scelta consente di:

- caricare una sola volta l'applicazione e aggiornare dinamicamente solo i componenti interessati;
- gestire la navigazione lato client tramite routing, mantenendo il contesto applicativo;
- integrare in modo naturale visualizzazioni dinamiche, tabelle reattive e la chat.[1, 20]

3.5.1 Struttura dell'applicazione

L'applicazione è organizzata in moduli principali:

- **layout e routing**: definiscono la struttura di base (header, sidebar, contenuto) e le rotte principali (dashboard, vista autore, pubblicazioni, chat);[20, 21]
- **componenti di dominio**: componenti dedicati a profilo autore, tabella pubblicazioni, schede venue e pannello della chat;
- **servizi di accesso alle API**: moduli che incapsulano le chiamate HTTP verso il backend (profilo, importazioni, chat, ecc.);
- **stato applicativo**: gestione dello stato condiviso (autore corrente, filtri attivi, sessione di chat selezionata).

La Figura 3.5 mostra la schermata principale della SPA, con il riepilogo delle informazioni principali contenute nel database in locale.

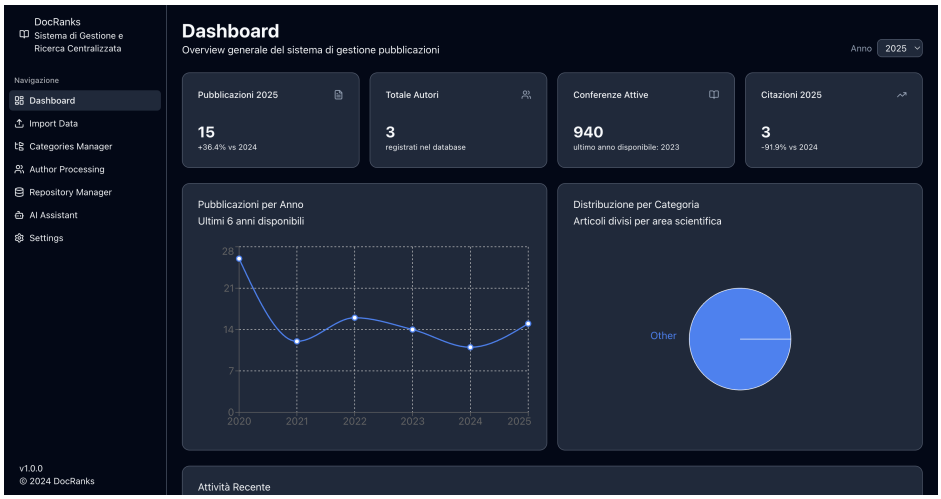

Figura 3.5: Schermata principale della SPA DocRanks 2.0 con riepilogo autore e lista delle pubblicazioni.

3.5.2 Interazione con le API e gestione dello stato

La SPA interagisce con il backend tramite un layer di servizi che espone funzioni ad alto livello, come:

- `getAuthorProfile(authorId)`;
- `getPublications(authorId, filters)`;
- `startImport(authorId)`;
- `listChatSessions()`, `sendChatMessage(sessionId, message)`.

Lo stato applicativo è gestito a livello di componenti e, dove necessario, tramite context, in modo da:

- mantenere coerenti i filtri tra tabella delle pubblicazioni e grafici;
- preservare la sessione di chat durante la navigazione;
- evitare richieste duplicate quando le informazioni sono già disponibili in memoria.

Il Listato 3.3 riporta uno pseudocodice rappresentativo del componente `RepositoryManager`, che gestisce stato locale, caricamento asincrono dei dati tramite API REST e rendering condizionale (loading/errore) della tabella.

Listing 3.3: Pseudocodice del componente React per gestione repository e rendering tabellare via API.

```
component RepositoryManager:
  state: items[], stats, loading, error, tab, query

  onMount:
    loading = true
    parallel:
      GET /repository/articles/list -> items.articles
```

```
      GET /repository/conferences/list -> items.
          conferences
      GET /repository/papers/list -> items.papers
      GET /dashboard/summary (+ /total-citations) ->
          stats
    loading = false

  view:
    if error -> show Alert(error)
    show Tabs(tab)
    show SearchInput(query)

    if loading -> show Spinner
    else -> show Table(items[tab] filtered by query)

  actions:
    delete(itemId):
      POST /repository/{type}/delete
      reload list + stats
```

3.6 Integrazione della Chat AI

Una delle principali estensioni funzionali introdotte in DocRanks 2.0 è l'integrazione di un assistente conversazionale basato su OpenAI, progettato come supporto alla consultazione e all'interpretazione dei dati bibliometrici.[6]

3.6.1 Architettura della chat

Dal punto di vista architetturale, la chat è implementata interamente lato backend:

- la SPA invia i messaggi dell'utente a endpoint dedicati;

- il backend costruisce il contesto a partire dai dati presenti nel database (profilo, pubblicazioni, venue) e dalla cronologia della sessione;
- il backend invia la richiesta al provider OpenAI e ne riceve la risposta;
- messaggi e risposte vengono salvati nel database e poi restituiti alla SPA.[6, 8]

Questa soluzione garantisce che la chiave API non sia mai esposta al client e che ogni interazione sia tracciabile e ripetibile.

Il Listato 3.4 mostra uno pseudocodice semplificato del servizio backend incaricato di gestire un nuovo messaggio di chat.

Listing 3.4: Servizio backend per la gestione di un messaggio di chat.

```
async def handle_chat_message(session_id: str,
    user_message: str) -> ChatMessage:
      history = repository.get_chat_history(session_id)
      context = build_context_from_db(session_id)
      response = openai_client.chat_completion(context,
          history, user_message)
      repository.save_messages(session_id, user_message,
          response)
      return response
```

3.6.2 Gestione del contesto e delle sessioni

Le conversazioni sono organizzate in *sessioni* di chat, ciascuna caratterizzata da:

- un identificativo univoco;
- un autore di riferimento e contesto associato (profilo, filtri correnti, vista attiva);
- una sequenza ordinata di messaggi utente/sistema.

La persistenza delle sessioni consente:

- di riprendere conversazioni precedenti mantenendo il contesto;
- di analizzare a posteriori le interazioni per scopi di audit e debugging;
- di evitare la ricostruzione manuale dei dati necessari a ogni nuova richiesta.[8]

3.6.3 Interrogazione del database tramite function calling (MCP)

Per rendere la chat utile anche per analisi e consultazioni non previste esplicitamente dall'interfaccia, DocRanks 2.0 adotta un approccio basato su *function calling*: il modello può richiedere l'esecuzione di tool lato backend per ottenere dati strutturati dal database.[6]

In particolare, il backend fornisce:

- una funzione per eseguire query `SELECT` (`execute_sql_query`) con vincoli di sicurezza (blocco di comandi non ammessi e aggiunta automatica di `LIMIT`);
- shortcut per statistiche ricorrenti (ad esempio conteggi autori, distribuzioni per categorie e ricerca pubblicazioni per autore).

Il flusso esecutivo è a due fasi: prima il modello propone eventuali tool call, poi il backend esegue le funzioni richieste e reinvia i risultati al modello, che produce una risposta finale in linguaggio naturale. Questo consente di mantenere i dati "sotto controllo" del backend, evitando accesso diretto dal client al database e preservando tracciabilità e consistenza delle risposte.[8]

3.7 Deployment e riproducibilità

Per facilitare lo sviluppo e la consegna del progetto, l'intero stack è stato containerizzato tramite Docker Compose.[7]

La composizione include:

- un container per MySQL con volume persistente;
- un container per il backend FastAPI eseguito su server ASGI;
- un container per il frontend React, servito come asset statici.[17, 18]

L'ambiente può essere avviato con un singolo comando:

```
docker compose up --build
```

rendendo il sistema riproducibile su macchine diverse e riducendo la complessità di configurazione manuale.[7]

3.8 Problemi incontrati e soluzioni adottate

Durante lo sviluppo di DocRanks 2.0 sono emerse alcune criticità ricorrenti, legate in particolare alle API esterne, alle performance su dataset numerosi e alla normalizzazione delle venue.[3, 4]

Rate limit delle API Scopus

Le API di Scopus impongono limiti stringenti sul numero di richieste giornaliere; per mitigarne l'effetto sono state adottate:

- politiche di caching dei risultati nel database locale;
- riduzione sistematica delle chiamate duplicate;
- strategie di *retry* controllato con backoff esponenziale.[3, 8]

Performance su dataset numerosi

Per autori con molte pubblicazioni l'interfaccia deve gestire tabelle estese senza degradare l'esperienza utente; le soluzioni adottate includono:

- paginazione lato backend con payload ridotti;
- filtri e ordinamenti eseguiti dal backend;
- rendering efficiente lato React, evitando ricalcoli non necessari.[11, 21]

Normalizzazione delle venue

La riconciliazione tra venue provenienti da fonti eterogenee ha richiesto un approccio a livelli:

- normalizzazione deterministica delle stringhe (trim, case, punteggiatura);
- uso preferenziale di identificativi affidabili quando disponibili;
- gestione esplicita di stati di incertezza (match ambigui o assenti) anziché inferenze non verificabili.[4]

3.9 Panoramica delle schermate della SPA

In questa sezione vengono presentate le principali schermate dell'applicazione DocRanks 2.0, ad esclusione della dashboard già illustrata in precedenza. Le immagini riportate mostrano le funzionalità operative della piattaforma dal punto di vista dell'utente finale.

3.9.1 Ricerca e gestione degli autori

La schermata di consultazione degli autori (Figura 3.6) permette di visualizzare l'elenco degli autori presenti nel database e di accedere rapidamente al profilo dettagliato di ciascuno.

Dal profilo autore è possibile accedere a una vista dettagliata delle metriche, delle pubblicazioni associate e delle informazioni aggregate (Figure 3.7 e 3.8).

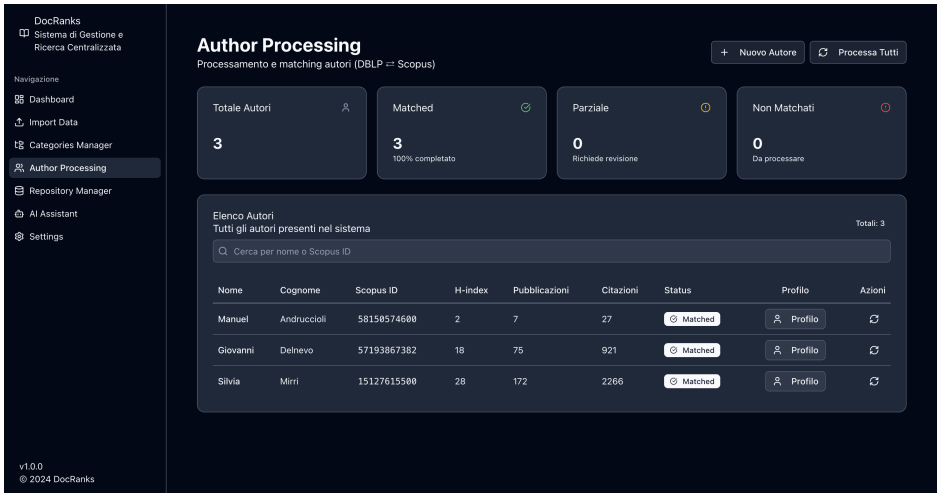

Figura 3.6: Schermata di consultazione degli autori presenti nel sistema.

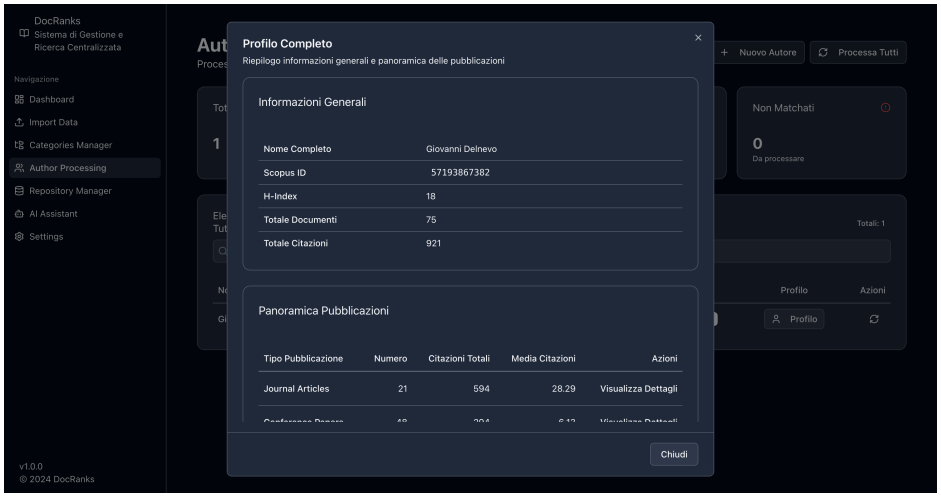

Figura 3.7: Profilo autore con metriche aggregate e riepilogo delle pubblicazioni.

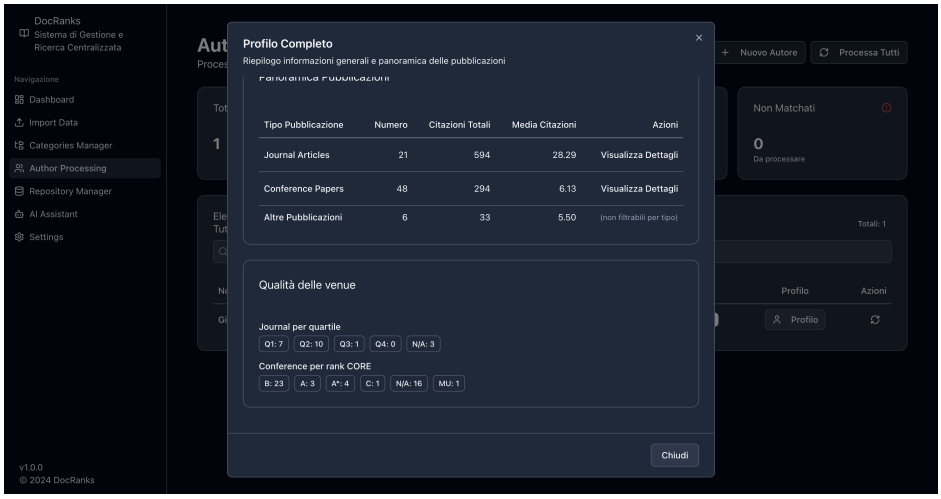

Figura 3.8: Dettaglio esteso del profilo autore e visualizzazione delle pubblicazioni.

3.9.2 Importazione e configurazione

La piattaforma consente l'importazione di nuovi autori e dataset tramite interfacce dedicate. La Figura 3.9 mostra la schermata di caricamento e gestione degli import.

Sono inoltre disponibili schermate di configurazione generale del sistema, come illustrato in Figura 3.10.

3.9.3 Gestione delle categorie e classificazioni

DocRanks 2.0 permette la consultazione e gestione delle categorie e delle classificazioni delle venue, come mostrato in Figura 3.11.

3.9.4 Chat AI integrata

Una delle principali estensioni funzionali della nuova versione è la Chat AI integrata. La Figura 3.12 mostra l'interfaccia conversazionale che consente di interrogare il database in linguaggio naturale.

L'integrazione della chat completa l'esperienza utente, affiancando alla navigazione tradizionale una modalità di consultazione guidata e dinamica.

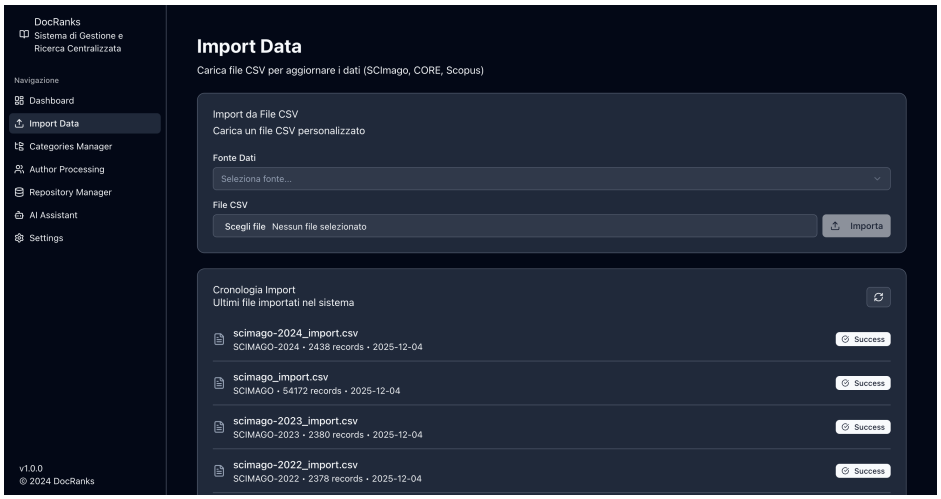

Figura 3.9: Schermata di importazione dati e gestione dei caricamenti.

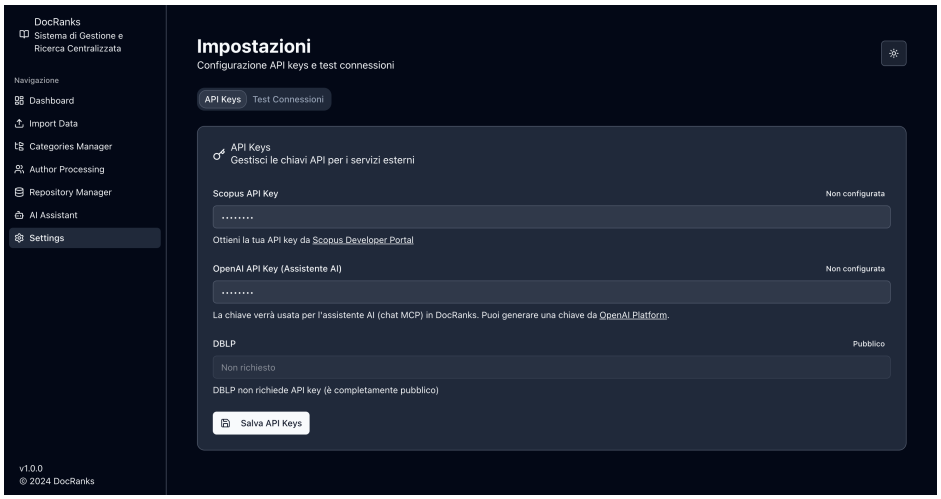

Figura 3.10: Schermata di impostazioni e configurazione del sistema.

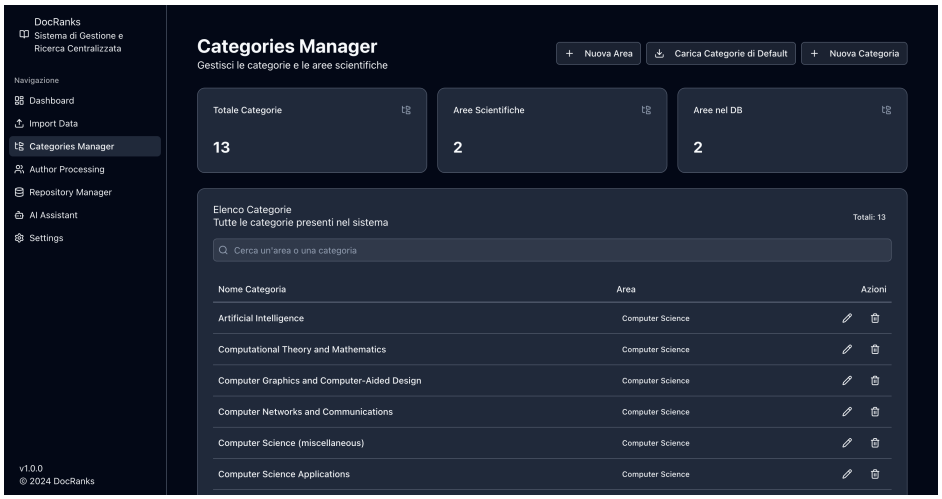

Figura 3.11: Schermata di consultazione delle categorie e classificazioni delle venue.

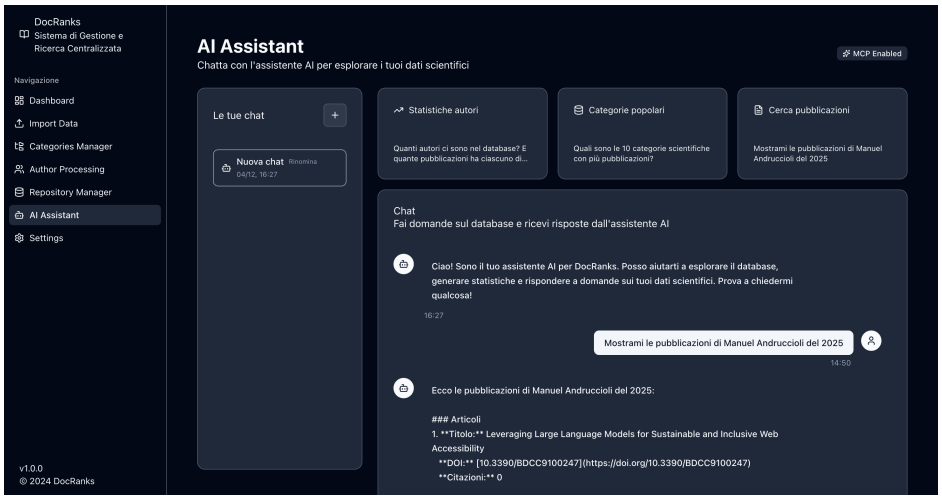

Figura 3.12: Interfaccia della Chat AI per interrogazione conversazionale del database.

3.10 Sintesi

DocRanks 2.0 concretizza gli obiettivi definiti nei capitoli precedenti traducendo le scelte architetturali in una piattaforma modulare, estendibile e riproducibile, capace di integrare fonti bibliometriche eterogenee e di supportare l'utente nella consultazione attraverso una SPA moderna e una Chat AI integrata.[1, 6]

La separazione tra backend Python, frontend React e persistenza MySQL consente di evolvere in modo indipendente le diverse parti del sistema, mantenendo coerenza e tracciabilità dei dati in un contesto bibliometrico complesso e in continua evoluzione.[8, 11, 12]

Conclusioni

La presente tesi ha affrontato il problema della frammentazione dei dati bibliometrici, proponendo e realizzando DocRanks 2.0 come sistema di integrazione orientato alla consultazione coerente e verificabile delle informazioni provenienti da fonti eterogenee.

Nel Capitolo 1 è stato analizzato il contesto bibliometrico, evidenziando le differenze tra database generalisti, portali specialistici e strumenti di classificazione qualitativa. È emerso come la distribuzione delle informazioni tra piattaforme differenti comporti un costo operativo significativo e renda complessa la costruzione di un quadro sintetico e affidabile del profilo di un autore.

Nel Capitolo 2 il problema è stato formalizzato in termini di frammentazione dei dati, dell'interazione e della semantica, definendo obiettivi funzionali e non funzionali coerenti con le esigenze di integrazione, tracciabilità e robustezza. È stata quindi delineata un'architettura logica basata su separazione delle responsabilità, persistenza locale e riduzione dell'accoppiamento tra componenti.

Nel Capitolo 3 tali principi sono stati tradotti in una realizzazione concreta: backend API-centrico in Python con FastAPI, organizzato in layer distinti; frontend SPA in React per una navigazione fluida e progressiva; modello dati relazionale progettato come *single source of truth* interna; pipeline ETL per l'importazione e la normalizzazione di dati provenienti da Scopus, DBLP, SCImago e CORE; integrazione di una Chat AI con accesso controllato al database tramite *function calling*.

I risultati ottenuti dimostrano che un livello di integrazione ben progettato può ridurre in modo significativo la frammentazione operativa, migliorando la coerenza della consultazione e mantenendo al contempo la tracciabilità delle informazioni rispetto alle fonti di origine. La trasformazione da applicazione monolitica a sistema modulare ha inoltre incrementato manutenibilità, estendibilità e riproducibilità dell'ambiente.

Tra i possibili sviluppi futuri si possono individuare:

- estensione del sistema a ulteriori fonti bibliometriche o dataset open data;
- miglioramento delle strategie di matching delle venue tramite tecniche più avanzate di similarità o supporto semantico;
- introduzione di funzionalità comparative tra più autori o gruppi di ricerca;
- evoluzione della Chat AI verso modalità di analisi più strutturate, con generazione automatica di report sintetici;
- approfondimento di aspetti di performance e scalabilità in scenari con dataset di dimensioni maggiori.

In conclusione, DocRanks 2.0 rappresenta un'evoluzione significativa rispetto alla versione originaria, non solo dal punto di vista tecnologico ma anche metodologico. L'integrazione coerente di fonti eterogenee, la separazione chiara delle responsabilità e l'attenzione alla tracciabilità costituiscono elementi centrali per la costruzione di strumenti affidabili in ambito bibliometrico. Il lavoro svolto dimostra come un progetto di integrazione, se guidato da principi architetturali solidi, possa trasformare un insieme frammentato di dati in una piattaforma unificata, consultabile e orientata al supporto decisionale.

Bibliografia

[1] Adobe. Single-page applications (spas) — what they are and how they work. https://business.adobe.com/blog/basics/learn-the-benefits-of-single-page-apps-spa, 2023.

[2] Mozilla Developer Network. Rest — representational state transfer. https://developer.mozilla.org/en-US/docs/Glossary/REST, 2024.

[3] Elsevier Developers. Scopus apis — developer documentation. https://dev.elsevier.com/api_docs.html, 2024.

[4] DBLP Team. Dblp api and dataset faq. https://dblp.org/faq/13501473.html, 2024.

[5] Armen Yuri Gasparyan, Lilit Ayvazyan, and George D. Kitas. Multidisciplinary bibliographic databases. *Journal of Korean Medical Science*, 28(9):1270, 2013.

[6] OpenAI. Openai api reference. https://platform.openai.com/docs/api-reference/introduction, 2024.

[7] Docker Inc. Docker compose documentation. https://docs.docker.com/compose/, 2024.

[8] MySQL. Mysql reference manual. https://dev.mysql.com/doc/, 2024.

[9] Ramotion. Spa vs mpa web architecture: Comparison & practices. https://www.ramotion.com/blog/spa-vs-mpa/, 2024.

[10] S. Gupta. Single page application (spa): What's so good (or bad)? https://www.jellyfishtechnologies.com/single-page-application-spa/, 2020.

[11] React Team. React documentation. https://react.dev/, 2024.

[12] Sebastián Ramírez. Fastapi documentation. https://fastapi.tiangolo.com/, 2024.

[13] Oracle Corporation. Mysql connector/python developer guide. https://dev.mysql.com/doc/connector-python/en/, 2024.

[14] SQLAlchemy Project. Dbapi and raw sql tutorial. https://docs.sqlalchemy.org/en/20/tutorial/dbapi_transactions.html, 2024.

[15] Pydantic Team. Pydantic documentation. https://docs.pydantic.dev/, 2024.

[16] ASGI Working Group. Asynchronous server gateway interface (asgi) specification. https://asgi.readthedocs.io/en/latest/, 2024.

[17] Uvicorn. Asgi server documentation. https://www.uvicorn.org/, 2024.

[18] Vite Contributors. Vite — next generation frontend tooling. https://vitejs.dev/, 2024.

[19] TailwindCSS Team. Tailwindcss documentation. https://tailwindcss.com/docs, 2024.

[20] React Router Contributors. React router documentation. https://reactrouter.com/, 2024.

[21] University of Washington. Client-side routing (info 340 course notes). https://info340.github.io/client-side-routing.html, 2024.

[22] Aiomysql Contributors. aiomysql documentation. https://aiomysql.readthedocs.io/, 2024.

[23] Facebook Inc. Jsx specification. `https://facebook.github.io/jsx/`, 2024.

[24] PyMySQL Team. Pymysql documentation. `https://pymysql.readthedocs.io/`, 2024.

[25] Shadcn. shadcn/ui documentation. `https://ui.shadcn.com/`, 2024.

[26] Astral Software. uv — python project and package manager. `https://github.com/astral-sh/uv`, 2024.

[27] LaTeX Project. Latex. `https://www.latex-project.org/`, 2024.

Ringraziamenti

Desidero esprimere la mia più sincera gratitudine ai miei genitori, che con pazienza, sostegno costante e fiducia mi hanno accompagnato lungo tutto il percorso universitario, aiutandomi ad affrontare ogni sfida con determinazione credendo sempre nelle mie potenzialità.

Un ringraziamento speciale va a mia sorella Alice, una presenza per me veramente preziosa: il suo supporto emotivo, i suoi consigli di esperienza e la sua vicinanza quando ne avevo più bisogno hanno rappresentato un punto di riferimento in tutti i momenti di difficoltà.

Desidero inoltre rivolgere un pensiero a Viviana che, pur avendo preso strade diverse, mi ha sostenuto in molti momenti importanti: ha saputo incoraggiarmi quando ero giù di morale, farmi sorridere quando ne avevo più bisogno e darmi la forza per proseguire verso i miei obiettivi. Mi sarebbe piaciuto condividere con lei anche questo traguardo, così come abbiamo condiviso molti dei nostri successi in passato, ma grazie a questa dedica è un pò come se fosse qui con me.

Ringrazio infine i miei relatori e correlatori per la disponibilità, la guida e il supporto che hanno dimostrato durante la realizzazione di questo progetto.