POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Reingegnerizzazione di un applicativo web per la gestione di una Biobanca dati

Relatore
Prof. Elena M. BARALIS

Corelatore
PhD. Alessandro FIORI

Candidato
Matteo ARZANI

Anno accademico 2017-2018

A mio Nonno Aldo,

che da lassù ancora mi insegni.

Sommario

Sparsi per il mondo esistono diversi istituti e centri di ricerca sul Cancro, come l'Istituto di Candiolo - IRCCS presso il quale questo progetto di tesi è stato sviluppato, le cui attività principali portano a produrre una enorme quantità di dati clinici, molecolari e biologici i quali richiedono una particolare gestione. I principali software destinati a fornire un supporto alle operazioni di laboratorio disponibili in commercio sono generalmente molto costosi e poco si adattano ai reali requisiti specifici di ogni differente realtà istituzionale. Per questo motivo nacque nel 2011 dalla collaborazione tra il "DataBase and Data Mining Group" del Politecnico di Torino e l'Istituto per la Ricerca e la Cura del Cancro a carattere Scientifico (IRCCS) di Candiolo il Laboratory Assistant Suite (LAS): un applicativo caratterizzato da una architettura modulare in grado di soddisfare le reali necessità. Analizzando il contesto in cui viene applicato, ogni paziente assistito presso l'istituto può firmare un consenso informato che autorizza il trattamento dei propri dati personali, degli esami effettuati e dei campioni biologici estratti a seguito di interventi chirurgici per scopi di ricerca. In particolare dai campioni vengono generate aliquote che possono essere utilizzate per esperimenti con le linee cellulari, impiantate in cavie animali con sistema immunitario compromesso (xenopazienti) o raccolte in contenitori il cui insieme complessivo rappresenta una Biobanca dati. Il software supporta i ricercatori durante tutte le attività svolte permettendo sia la registrazione dei dati sia l'analisi di questi in maniera tale da estrarne preziose informazioni e conoscenza medico/scientifica con la finalità di determinare nuove terapie efficaci per la cura delle neoplasie.

Il LAS è progettato come una architettura su più livelli intesi come quattro macro-moduli principali (gestionale, operativo, integrativo e analitico) ognuno a sua volta costituito da sotto-moduli. L'applicazione è caratterizzata da una parte server che espone un servizio sviluppato utilizzando il framework Django il quale interagisce con tre differenti database (MySQL, MongoDB e Neo4J) e da una parte client che si basa su HTML, JavaScript e jQuery. Questo progetto ha dimostrato alcune limitazioni, in parte dovute alla sua specifica struttura in parte dovute all'avvento di moderne tecnologie. Il server non rispetta il modello di maturità proposto da Richardson e non può essere considerato un vero e proprio servizio REST, inoltre, il tasso di riutilizzo del software, nonostante l'architettura modulare, è relativamente basso e la gestione di tre differenti basi di dati ha aumentato la complessità del sistema e della sua gestione. Il contenuto client viene generato dal server su richiesta, mentre oggi è possibile creare applicazioni a pagina singola le quali spostano parte della logica di business e gestiscono le interfacce direttamente sul browser del dispositivo che si connette al servizio permettendo lo sviluppo di una applicazione più efficiente

e fluida. Infine, l'analisi di alcuni report da parte degli utilizzatori abituali del software ha fatto emergere alcune criticità relative al suo impiego.

Sulla base di queste considerazioni e con l'idea di creare un servizio specifico dedicato alla gestione di una Biobanca dati, prevista precedentemente solo come un semplice sotto-modulo, è nato questo progetto di reingegnerizzazione del software. In particolare nel nuovo Framework sono state previste tre applicazioni principali: un Blog divulgativo per raggiungere i cittadini e informarli delle attività svolte e dei progressi compiuti in ambito scientifico, una Biobanca dati in grado di collezionare e permettere la gestione, nonché la condivisione con altre realtà, di tutti i dati biologici raccolti e, per finire, una versione innovativa del precedente progetto chiamata LAS 2.0. Sono state valutate diverse soluzioni e diverse tecnologie del panorama attuale. Il backend è stato sviluppato utilizzando sempre Django Framework ma creando un servizio REST, il quale espone diverse API, orientato al modello architetturale basato su micro-servizi. Per quanto riguarda il database è stato utilizzato unicamente MongoDB organizzato in replica-set per permettere una distribuzione dei dati su più nodi; la versione utilizzata, la 4.0, permette di usufruire anche di un supporto alle transazioni ACID e altri utili strumenti che lo rendono decisamente una delle migliori scelte possibili ad oggi. Il frontend costituito dai tre applicativi, invece, è stato così strutturato: un semplice Blog basato sul CMS Wordpress e due applicazioni a pagina singola deposte su domini differenti sviluppate utilizzando Angular Framework. Per garantire una buona manutenibilità del software e un meccanismo di riutilizzo del codice, sia il server che il client sono stati programmati basandosi sul concetto di modularità. Per quanto riguarda il web server è stato utilizzato NGINX con lo scopo di garantire efficienza e scalabilità al servizio offerto. Il progetto è stato sviluppato anche con l'obiettivo di permetterne la distribuzione ad altri istituti che possano così personalizzare i contenuti e accedere a ogni sua funzionalità. Per questo motivo e, in ottica di ottimizzare le risorse hardware disponibili e di distribuire il servizio su più nodi, è stata utilizzata la piattaforma Docker con particolare attenzione a Docker Swarm che permette la virtualizzazione in ambito cloud.

Durante la creazione di questo progetto sono state poste le basi per il funzionamento dell'intero sistema. Uno degli aspetti principali sviluppati è stato il processo di autenticazione, opportunamente diversificato per la Biobanca e per il LAS 2.0 a seconda dei casi d'uso. La prima, infatti, prevede l'iscrizione di utenti singoli anche al di fuori della realtà istituzionale mentre per il secondo, limitatamente all'istituto, il processo di registrazione può coinvolgere un intero gruppo di lavoro (opportunamente sottomesso dal responsabile) o solamente un singolo operatore il quale richiede così di poter lavorare per un gruppo. Entrambi i processi di login si basano sul meccanismo classico di utilizzo del binomio username e password. L'autenticazione è stata sviluppata principalmente con uno scopo funzionale in modo tale da fornire un primo supporto per l'utilizzo e il test dell'applicativo. Non sono stati richiesti interventi tecnici o progettuali specifici relativi alla sicurezza e sono state quindi utilizzate le principali soluzioni tecnologiche predisposte e consigliate dal team IT, tuttavia, è stata effettuata un'analisi di alcuni aspetti sottolineando vantaggi e svantaggi delle tecniche adottate.

Un altro aspetto fondamentale definito è quello che riguarda la gestione dei privilegi. Come prima semplice suddivisione sono stati identificati due ruoli principali

di utente: ADMIN e GUEST. Inoltre, è stata modellata una struttura dei permessi associati ai dati basata sul concetto di etichette. Sostanzialmente, ogni bioentità raccolta e memorizzata nel database possiede determinate regole di accesso e regole di propagazione dei permessi che le entità figlie ereditano. Le etichette, relative per esempio a un gruppo di lavoro piuttosto che a un progetto o altre entità, permettono di fornire le informazioni per discriminare se un utente può accedere o meno al dato. Inoltre, sulla base dell'architettura del client, è stato necessario creare un meccanismo in grado di mostrare ad ogni utente solo l'insieme di interfacce (raccolte sotto il nome di funzionalità) per cui è autorizzato, evitando quindi di permettere la visualizzazione di contenuto non utilizzabile dato il livello di privilegio.

Altre principali innovazioni sono state la gestione di uno stato lato client grazie al pattern di programmazione Redux e la gestione dei dati in arrivo dal server mediante paginazione. Sono stati sviluppati un modulo Amministrativo e un modulo Profilo Utente. Il primo è in grado di permettere la gestione delle richieste di registrazioni provenienti dal LAS 2.0 e la concessione di nuovi permessi e funzionalità agli utenti del sistema, il secondo, invece, permette di visualizzare i dati relativi al profilo e, in modo duale, di richiedere una estensione dei privilegi. Un altro modulo sviluppato è quello relativo alla messaggistica, il quale permette la comunicazione tra gli utenti in maniera simile al modello classico delle email ed è previsto anche un modulo JSON in grado di gestire la visualizzazione e la modifica di contenuto omonimo. Infine, è stato sviluppato un servizio di cache per permettere il salvataggio di informazioni nell'IndexedDB del Browser Web. Il suo scopo è quello di fornire una base futura per ogni modulo operativo del sistema in grado di permettere il funzionamento degli applicativi anche in assenza di rete e lavorando su copie locali dei dati.

L'intero progetto è stato sviluppato ponendo particolare attenzione al processo di documentazione. In particolare al servizio REST è stata associata la tecnologia Swagger la quale ha permesso di creare una pagina in cui vengono mostrate tutte le API con relative azioni, parametri e struttura del corpo di richieste e risposte HTTP; per quanto riguarda la parte client, è stato creato un piccolo manuale contenente le principali linee guida per facilitare l'implementazione agli sviluppatori nel lungo periodo. Per migliorare la gestione complessiva dell'ambiente virtualizzato, sia in fase di sviluppo sia in fase di produzione, è stato adottato Portainer: uno strumento che fornisce una interfaccia grafica user-friendly compatibile con la piattaforma Docker.

Durante la progettazione sono state prese in considerazione e analizzate le principali normative europee in ambito di privacy e gestione dei dati per le biobanche, le quali, certamente, sono oggetto del lavoro svolto dal team legale dell'istituto ma rappresentano la base fondante per la progettazione di un database e di un applicativo conforme con la legge.

Allo stato attuale il progetto rappresenta un prototipo le cui tecnologie, architettura generale e moduli base sono stati definiti e testati. Gli obiettivi futuri, per permetterne la produzione, coinvolgeranno l'implementazione dei moduli operativi del LAS 2.0 e delle interfacce della Biobanca dati sulla base dei principali casi d'uso.

Ringraziamenti

Scrivere questo breve ringraziamento rappresenta l'ultimo capitolo della mia tesi. Un capitolo dove non si parla di software o tecnologie ma delle persone che sono state importanti durante il mio percorso, fatte di carne ed emozioni.

Questi cinque anni hanno rappresentato un periodo di profondo apprendimento, che ha portato una crescita non solo a livello scientifico, ma anche personale.

In ogni momento di difficoltà la mia mente mi riportava ad uno scenario dove mi immaginavo di dover abbattere un albero di cui non si vedeva né l'inizio né la fine. In questi anni non mi sono perso d'animo e ho trovato la forza e la volontà di abbatterlo, ho affilato la scure e ho colpito più volte, sempre nello stesso punto fino a raggiungere il mio obiettivo. Se ci sono riuscito è anche merito di tutte quelle persone che mi han donato la forza necessaria in ogni momento, la forza per ogni colpo.

Vorrei quindi spendere due parole di ringraziamento nei confronti di tutte le persone che mi hanno sostenuto e aiutato durante questo periodo.

Prima di tutto intendo ringraziare la mia Relatrice nonché Professoressa Elena Maria Baralis che tanto mi ha insegnato della sua disciplina e il Corelatore PhD. Alessandro Fiori il quale ha dimostrato fin da subito una grande disponibilità nei miei confronti e una grande passione per il nostro lavoro diventando quello che io definisco il mio mentore.

Ringrazio il Professore Giovanni Malnati per tutta la conoscenza che mi ha trasmesso e per le parole spese in un discorso finale di uno dei corsi da lei tenuto, ricorderò sempre le sue parole.

Ringrazio il Professore Antonio Servetti per i consigli e la disponibilità dimostratami.

Vorrei ringraziare l'Istituto di Candiolo - IRCCS e in particolare il Dott. Andrea Bertotti e i colleghi con cui ho lavorato l'Ing. Andrea Mignone e l'Ing. Massimiliano Frassà.

Infine desidero ringraziare tutta la mia famiglia, in special modo i miei genitori e i miei nonni che mi hanno sostenuto sia economicamente sia moralmente giorno dopo giorno, i miei cari amici presenti in ogni momento sia esso di gioia sia esso di dolore e la mia ragazza che, beh si sà, tanta pazienza porta per stare con un Ingegnere.

Un sentito grazie a tutti, un abbraccio di cuore!

Indice

Capitolo 1

Introduzione

Oggi nell'era post-genomica, con l'avvento delle nuove tecnologie viene generata un'enorme quantità di dati molecolari complessi ad alto rendimento.
La gestione di dati clinici, molecolari e biologici è sicuramente un compito impegnativo, a causa della complessità e dell'eterogeneità dei dati. Inoltre, l'analisi di dati molecolari eterogenei, integrati con dati clinici e informazioni biologiche, è un prerequisito importante per scoprire le cause principali dei tumori. Infatti, i dati genomici codificano molte intuizioni biologiche di cui l'indagine può essere la base per ottenere progressi nel campo dell'oncologia molecolare. Molti sforzi sono stati dedicati alla costruzione di sistemi di gestione per questo tipo di dato [1].

1.1 Nascita del progetto

Una delle principali tecnologie ICT impiegate nei laboratori di ricerca è il Laboratory Information Management Systems (LIMS). Parte di questi ultimi sono disponibili sul mercato, ma, di solito, sono molto costosi e richiedono una notevole quantità di risorse umane ed economiche per adattarsi ai requisiti specifici di laboratorio [2].

Per superare i limiti delle soluzioni proposte sul mercato e contenere i costi relativi alle risorse impiegate in questo ambito nacque nel 2011, dalla collaborazione tra il "DataBase and Data Mining Group" del Politecnico di Torino e l'Istituto per la Ricerca e la Cura del Cancro a carattere Scientifico (IRCCS) di Candiolo, il Laboratory Assistant Suite (LAS). Questo applicativo assiste i ricercatori durante le loro attività di laboratorio. La sua architettura modulare consente di gestire diversi tipi di dati grezzi (ad es. biologici, molecolari) e dati sperimentali di tracciamento. Inoltre, la piattaforma supporta l'integrazione di diverse risorse e aiuta a eseguire una varietà di analisi al fine di estrarre conoscenze relative ai tumori. Le interfacce utente vengono progettate per consentire la gestione dei dati in ambienti ostili in cui i ricercatori devono minimizzare le loro interazioni con il sistema (ad es. in condizioni sterili).

I progressi in campo biomedico hanno focalizzato l'attenzione della comunità di ricerca internazionale sull'analisi dei dati raccolti durante le operazioni di laboratorio per una comprensione più profonda. Queste raccolte possono essere definite biobanche e contengono al loro interno le informazioni relative ai biomateriali di pazienti

e/o cavie da laboratorio, nonché, studi e analisi effettuate applicando la conoscenza medica e scientifica.

1.2 Istituzioni coinvolte

Candiolo Cancer Institute è un'istituzione privata senza fini di lucro fondata e sostenuta dalla Fondazione Piemontese per la Ricerca sul Cancro-Onlus (FPRC) e gestita dalla Fondazione del Piemonte per l'Oncologia (FPO: cofondata da FPRC e Regione Piemonte). L'Istituto è riconosciuto dall'IRCCS (Istituto di Ricovero e Cura a Carattere Scientifico) dal Ministero della Salute italiano. Candiolo Cancer Institute lavora in sinergia con la Scuola Medica dell'Università di Torino.
La sua missione è un contributo significativo per combattere il cancro, comprendendo le basi e fornendo servizi diagnostici e terapeutici all'avanguardia. L'interfaccia tra biologia molecolare e medicina è al centro dell'Istituto. L'FPRC fornisce una raccolta permanente di fondi per completare e sviluppare gli edifici e le tecnologie dell'Istituto al fine di promuovere la ricerca [40].

Questa istituzione è da diversi anni impegnata nell'ambito della ricerca nella lotta contro i tumori e precisamente possiede un LAS sviluppato sulla base delle necessità delle attività svolte dai ricercatori nei laboratori.
L'utilizzo di tecnologie obsolete, l'avvento di nuove tecnologie e la necessità di modernizzazione hanno dato vita a questo progetto il cui scopo fondamentale è quello di donare nuova vita al progetto iniziale del LAS integrandolo con un applicativo destinato alla gestione di una Biobanca dati che possa lavorare in sinergia con esso.

1.3 Obiettivo del progetto

L'obiettivo principale di questo progetto è quindi quello di reingegnerizzare il framework esistente, creando un applicativo costituito da un moderno LAS, un applicativo dedicato alla gestione di una Biobanca dati e un Blog utilizzato per la diffusione di notizie e informazioni al pubblico.
Il progetto, inoltre, potrà essere distribuito e reso adattabile eventualmente per altri istituti i quali potranno configurare e personalizzare il proprio Blog e utilizzare le funzionalità degli applicativi sviluppati.

Le principali operazioni svolte durante la creazione del progetto sono state:

1. Analisi e ispezione del framework 1.0 per evidenziarne i limiti e i punti deboli.

2. Confronto delle moderne tecnologie a disposizione per lo sviluppo di applicativi web.

3. Scelta dell'architettura per il nuovo framework, prevedendo quali componenti software del vecchio modello eventualmente riutilizzare e quali sostituire completamente.

4. Creazione di un nuovo framework basato sui concetti di modularità, riutilizzo del codice, efficienza, scalabilità e manutenibilità.
5. Organizzazione in maniera ottimale del progetto e fornitura di un supporto alla documentazione per permettere al team di sviluppo in futuro di poter effettuare aggiornamenti e modifiche al sistema in maniera immediata e semplice.
6. Sostituzione dei moduli di autenticazione e gestione dei permessi.
7. Analisi dei principali casi d'uso introducendo nuovi modelli e concetti per quanto riguarda la logica dell'applicativo.
8. Operazioni di debug e test preliminari effettuate sul prototipo sviluppato.

In particolare durante la fase implementativa sono state create le basi per la gestione dei tre applicativi principlai (Biobanca, Blog e LAS), ponendo particolare attenzione per alcuni aspetti che hanno permesso di sviluppare il prototipo iniziale:

- Autenticazione degli utenti (gestita in maniera differente per il dominio del LAS rispetto al dominio della Biobanca);
- Gestione dei permessi;
- Sviluppo di un supporto alla messaggistica;
- Creazione di un modulo di Amministrazione centralizzato;
- Sviluppo di un servizio di cache dei dati che permetta il loro salvataggio nell'IndexedDB del browser;
- Sviluppo di un modulo, disponibile all'amministratore, che permetta di accedere a un'interfaccia di gestione dei documenti salvati nel database in MongoDB;
- Creazione di componenti riutilizzabili per la paginazione dei dati e la loro tabulazione.

Capitolo 2

Contesto

Il LAS, come accennato nel Capitolo 1, fa parte della classe di software chiamata LIMS (Laboratory Management Information System). Questo tipo di applicativi mira a tracciare, integrare e a rendere fruibile l'informazione generata nei laboratori biomedici. La piattaforma è attualmente funzionante ed assiste quotidianamente i ricercatori nelle attività di laboratorio.
Il compito del LAS è quello di fornire le funzionalità normalmente messe a disposizione dai software LIMS, integrandole con strumenti di analisi che sfruttino tutti i più recenti studi di data mining e bioinformatica, per permetterne una corretta interpretazione dei dati e facilitare l'estrazione di informazioni utili per la ricerca contro il cancro.
Lo scopo finale di questa tecnologia è quindi quello di semplificare le procedure eseguite quotidianamente nei laboratori di ricerca, aiutare a compiere analisi accurate relative ai tumori e permettere di sviluppare nuove efficaci terapie.

2.1 Principali Attività

La Figura 2.1 mostra la pipeline delle principali attività svolte all'interno del centro. A seguito di interventi chirurgici eseguiti su pazienti, i quali hanno firmato un apposito consenso informato (vedi Capitolo 9), vengono raccolti campioni tumorali. Ogni campione biologico derivato dal singolo paziente permette la generazione di una serie di aliquote. In base alle caratteristiche delle aliquote e allo scopo dello studio di ricerca, le operazioni che possono essere eseguite sono:

- Stoccaggio in appositi contenitori (linee cellulari);
- Estrazione di aliquote derivate (ad esempio DNA o RNA);
- Impianto seriale in animali aventi sistema immunitario compromesso (ovvero xenopazienti);
- Registrazione del dato in formato digitale presso la Biobanca dati;
- Sperimentazioni e analisi.

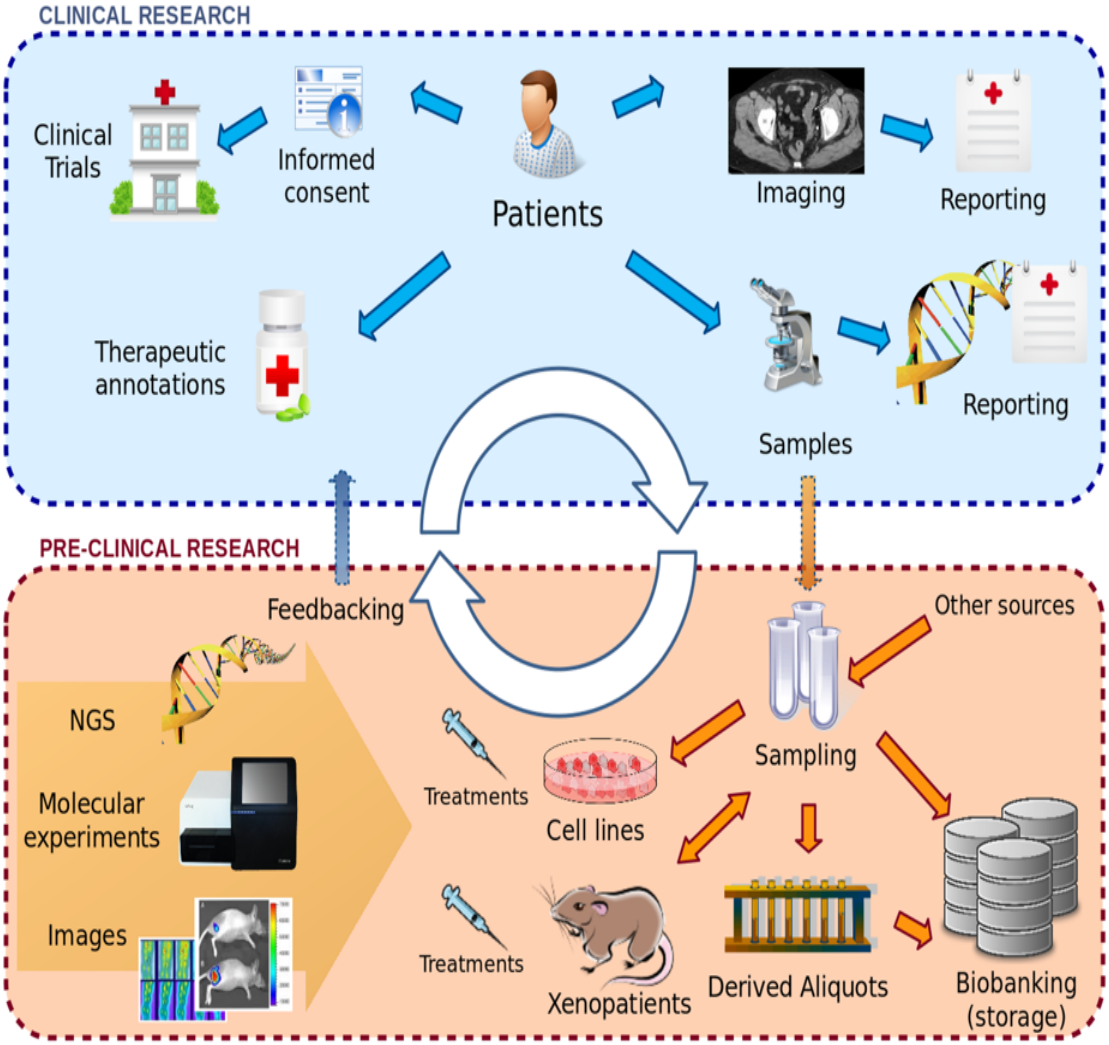

Figura 2.1: Principali attività del LAS

I ricercatori definiscono protocolli di trattamento per monitorare l'evoluzione della massa tumorale impiantata negli xenopazienti tramite misurazioni periodiche e valutano la loro risposta ai farmaci. Inoltre, possono generare nuovi campioni di tessuto provenienti da questi animali per ulteriori analisi. Infatti, tutte le aliquote derivate possono essere sfruttate per analisi sperimentali basate su diverse tecnologie.
I valori di espressione di migliaia di geni, ad esempio, possono essere analizzati mediante la tecnologia microarray. Diversamente, con le tecniche di sequenziamento di prossima generazione, è possibile estrarre e analizzare la sequenza completa del DNA di un campione.

L'esistenza di un unico sistema di riferimento per i ricercatori permette di semplificare i processi di laboratorio garantendo la loro interoperabilità.
L'obiettivo finale è quello di aiutare a tradurre le informazioni correlate ai dati clinicamente rilevanti e funzionalmente convalidati per determinare nuove terapie farmacologiche e cliniche efficaci che possano essere documentate correttamente e diventare, quindi, nuova conoscenza in campo medico e scientifico.

2.1.1 Gestione dei contenitori

Durante le attività di laboratorio vengono utilizzati diversi contenitori per raccogliere il materiale biologico. All'interno del LAS esistono diversi tipologie di contenitori:

1. freezer o armadietti a vetrina: sono i contenitori di dimensione massima;
2. cassetti o scaffali: sono solitamente i sotto-contenitori della prima categoria;
3. piastre: contengono direttamente il materiale biologico se formate da tubi;
4. tubi o cassette biologiche: contengono materiale biologico;

Alcuni di essi possono contenere materiale biologico altri un sotto-contenitore e così via, creando un albero di contenitori innestati. Per questo motivo vengono organizzati tenendo traccia di un identificativo univoco. In questa maniera il sistema software è in grado di tenere sotto controllo contenitori e materiali biologici raccolti, permettendo al ricercatore in laboratorio di ottenere facilmente ciò di cui ha bisogno durante lo svolgimento delle sue attività. In questo ambito esistono sostanzialmente due procedure generali: la prima permette all'operatore di reperire il contenitore per compiere determinati operazioni, la seconda consiste nel ricollocare correttamente il materiale utilizzato.

2.1.2 Gestione dei biomateriali

Attualmente all'interno del LAS è presente un "Modulo Biobanca" il quale permette di raccogliere e salvare le informazioni relative ai biomateriali [1]. In base ai tipi di materiali biologici che immagazzinano, le biobanche sono comunemente suddivise in biobanche di tessuto e genetiche. Il LAS attuale si fa carico di gestire la memorizzazione di entrambe le tipologie di informazione, compresa la gestione di campioni biologici e dati provenienti da strumenti per la misurazione, nonché il supporto a una serie di procedure relative al laboratorio.

Le aliquote raccolte dall'intervento chirurgico e le aliquote derivate sono gestite separatamente poiché le relative informazioni e procedure sono diverse. Il ricercatore, in ogni passaggio delle derivazioni, viene assistito dallo strumento software. La composizione di ciascuna aliquota derivata è suggerita in base al tipo di derivazione (ad es. DNA, RNA) e alle regole di laboratorio sul volume e sulla concentrazione della miscela ottenuta. Inoltre, vengono tracciati tutti i kit utilizzati durante le suddette procedure di derivazione per permettere una completa e trasparente gestione del magazzino e per monitorare gli indici di qualità associati ad ogni kit e ogni aliquota derivata. La qualità, infatti, è uno degli elementi più influenzanti durante l'esecuzione degli esperimenti di laboratorio.

Per consentire l'impiego di lettori di codici a barre e quindi semplificare le procedure di immissione dei dati, la maggior parte dei dispositivi sperimentali (vale a dire cartelle cliniche, piastre, tubi) sono codificati a barre dall'istituto di ricerca prima del

loro utilizzo. Questo consente un profondo monitoraggio dell'intero sistema, ponendo attenzione non solamente ai campioni biologici raccolti ma anche agli strumenti utilizzati per raccoglierli e contenerli.

I ricercatori durante le analisi di laboratorio possono quindi memorizzare diversi oggetti biologici e molecolari. A questi è possibile associare altre informazioni (ad esempio annotazioni) raccolte da ulteriori analisi. In pratica, i ricercatori possono assegnare a un gruppo di oggetti la risposta a un trattamento farmacologico secondo un'analisi statistica eseguita su dati molecolari e biologici corrispondenti. Inoltre, i dati integrati possono essere sfogliati o visualizzati per mezzo di rappresentazioni grafiche (ad es. Alberi genealogici) o diagrammi (ad es. Diagrammi di Venn).

2.1.3 Gestione degli xenopazienti

Una delle attività cruciali predisposta dal LAS è la gestione degli animali immunocompromessi (cioè, xenopatienti). Risulta necessario monitorare il ciclo di vita di ogni xenopaziente, dalla loro acquisizione da parte dell'istituto di ricerca fino alla loro morte [1].

I topi impiegati dall'IRCCS sono dotati di etichette RFID, che consentono un'identificazione rapida e semplice di ciascun animale. La necessità di caricare nel sistema un'intera serie di cavie per volta beneficia di questa struttura identificativa. In base ai progetti di ricerca e delle attività di laboratorio stabilite, dopo il caricamento dei topi nella base dati associata al software, i frammenti di tessuto tumorale vengono impiantati negli xenopazienti disponibili. L'applicativo gestisce efficacemente questa operazione accedendo al contenuto della biobanca e recuperando le aliquote tumorali attualmente memorizzate nella piastra sorgente. La crescita del tumore all'interno dello xenopaziente viene monitorata mediante la misurazione qualitativa o quantitativa della massa tumorale.

Esistono quattro modalità principali di misurazione:

1. osservazione: la dimensione del tumore viene valutata sulla base delle dimensioni dell'animale con un range di valori prestabilito e universalmente riconosciuto (per esempio non presente, molto piccolo, piccolo, medio e grande);
2. peso: viene solamente pesato l'animale;
3. volume tumorale: tramire appositi strumenti viene valutato il volume dell'entità tumorale;
4. volume tumorale e peso: è la combinazione delle precedenti due.

I trattamenti, volti a valutare l'efficacia di determinati farmaci, possono essere assegnati a xenopatienti misurati. Ogni trattamento è composto da diverse fasi, associate a diversi farmaci. In generale, per ciascuna fase, è possibile definire il farmaco, la modalità di somministrazione, la dose e la frequenza di somministrazione. Inoltre, un diagramma di Gantt consente di specificare un orario per ciascuna fase. Il ciclo di vita di un generico xenopaziente è terminato da un espianto, il quale genera un

numero di aliquote di tessuto tumorale, rese disponibili per gli impianti futuri. Come per gli impianti, le operazioni di espianto sono coordinate con il modulo della biobanca per visualizzare le lastre di destinazione e trasmettere dati sulle aliquote generate.

2.1.4 Gestione degli esperimenti in vitro

La sperimentazione in vitro, ad esempio le linee cellulari, permettono la manipolazione dei tumori rappresentati da un aliquota in un sistema artificiale controllato in grado di testare gli effetti delle differenti terapie farmacologiche.
Per linea cellulare il LAS intende un insieme di entità biologiche generate dallo stesso individuo (xenopaziente) sotto le stesse condizioni di sperimentazione. Tali condizioni sono definite da opportuni protocolli che descrivono il tipo di processo e i parametri di coltura applicati. Allo stato attuale il software è in grado di gestire i protocolli e la generazione delle linee cellulari. Queste linee cellulari vengono gestite da gruppi di ricercatori e su di esse possono essere compiute diverse operazioni quali:

1. eliminare una o più piastre;
2. espandere le linee cellulari definendo parametri di diluizione, protocollo di espansione e numero di piastre in output al processo;
3. sottomettere una serie di piastre ad un esperimento;
4. pianificare e archiviare una procedura di sperimentazione.

2.1.5 Gestione degli esperimenti molecolari

I principali esperimenti molecolari gestibili all'interno del LAS sono:

- Reazione a catena della polimerasi digitale: comunemente nota con al sigla dPCR, è una tecnica che permette l'amplificazione o moltiplicazione dei frammenti di acidi nucleici quali DNA, cDNA e RNA sulla base delle loro sequenze nucleotidiche iniziali e terminali. Questo permette di ottenere in vitro il materiale genetico necessario per successive attività di laboratorio. Rispetto alla PCR tradizionale i campioni sono suddivisi in un maggior numero di partizioni in ognuna delle quali avviene la reazione base.
- Reazione a catena della polimerasi in tempo reale: rispetto alla precedente è un metodo che simultaneamente amplifica e quantifica il DNA.
- Irradiazione: questa tecnologia può essere utilizzata per rilevare le mutazioni delle cellule cancerogene senza la necessità di una biopsia fisica. In particolare rileva cellule tumorali circolanti (ctDNA) situate a basse concentrazioni in campioni di sangue. I ricercatori, da questa analisi, riescono a reperire fondamentali informazioni circa le fasi iniziali, lo sviluppo e la reazione della neoplasia alle terapie somministrate, evitando ogni tipo di intervento chirurgico invasivo.

- Microarray: è un insieme di microscopiche sonde di DNA ottenute dalla tecnica PCR disposte su di una superficie di vetro, plastica, o chip di silicio che per l'appunto formano una matrice (o array). Questa moderna sperimentazione permette di esaminare simultaneamente i geni del DNA, determinando eventualmente quali possono essere i geni coinvolti nella malattia.

- Sequenziamento del DNA: le tecniche principali sono il sequenziamento di Sanger, il quale è stato per 40 anni il metodo più usato o il sequenziamento in parallelo (in inglese, next generation sequencing), il quale rappresenta un insieme di tecnologie moderne che permettono di sequenziare in breve tempo numerose quantità di genomi.

- Sequenom: questa tecnologia, disponibile in commercio, offre una serie di test genetici accurati, affidabili e immediati dalla quale è possibile reperire informazioni sulle implicazioni genetiche della salute [4].

2.1.6 Analisi

Monitorare le attività svolte in laboratorio e raccogliere i dati legati ai campioni biologici è di fondamentale importanza, tuttavia i ricercatori hanno bisogno anche di un supporto per manipolare e integrare informazioni a carattere eterogeneo per determinare nuova conoscenza.

Il LAS attualmente predispone un insieme di strumenti in grado di pianificare opportune interrogazioni e analisi complesse. Sulla base dei risultati parziali o finali è possibile estrarre informazioni utili che vengono poi memorizzate nel sistema.
Attraverso questa attività vengono ottenuti risultati importanti i quali, spesso, vengono pubblicati su articoli e documenti a carattere scientifico.
L'obiettivo delle analisi permette una profonda valutazione del lavoro svolto nel laboratorio identificando l'efficacia delle terapie e facendo emergere nuove evidenze mediche.

Capitolo 3

Analisi dello stato dell'arte

3.1 Framework 1.0

Il framework 1.0 è costituito principalmente dal LAS: una piattaforma progettata per fornire assistenza a ricercatori di laboratori biologici e biomedici durante lo svolgimento delle loro principali attività.

3.1.1 Architettura

IL LAS è stato progettato come una architettura su più livelli (Figura 3.1). Questi livelli possono essere intesi come i quattro macro-moduli principali. Ognuno di essi ha un compito specifico e può eventualmente interagire con gli altri attraverso apposite interfacce e si compone a sua volta di sotto-moduli:

- Livello gestionale: le sue funzioni principali sono relative all'autenticazione degli utenti e alla gestione dei permessi all'interno dell'intero ambiente.
- Livello operativo: il suo compito è di ottenere, memorizzare e monitorare i dati relativi a diverse operazioni eseguite da fonti differenti (moduli). Le interfacce client di questo livello sono quelle esposte agli utilizzatori finali durante il lavoro nel laboratorio e si basano su aspetti quali l'usabilità e l'intuibilità, nonché la facilità di utilizzo nonostante l'ambiente ostile del laboratorio. E' prevista un'interazione con apparecchiature elettroniche di supporto quali scanner di codice a barre.
- Livello integrativo: questo livello permette di eseguire interrogazioni complesse sui dati raccolti. Le entità biologiche vengono associate ad appositi identificatori univoci per la loro memorizzazione nel database. I risultati ottenuti possono essere utilizzati dal livello di analisi per permetterne una annotazione definitiva (l'annotazione altro non è che l'esito dell'elaborazione associato all'entità su cui è stata effettuata tale operazione).
- Livello analitico: tramite apposite interfacce di analisi gli utenti possono progettare un workflow mettendo in cascata una o più entità da analizzare. Questo livello si appoggia a tecnologie esterne per compiere calcoli e studi approfonditi.

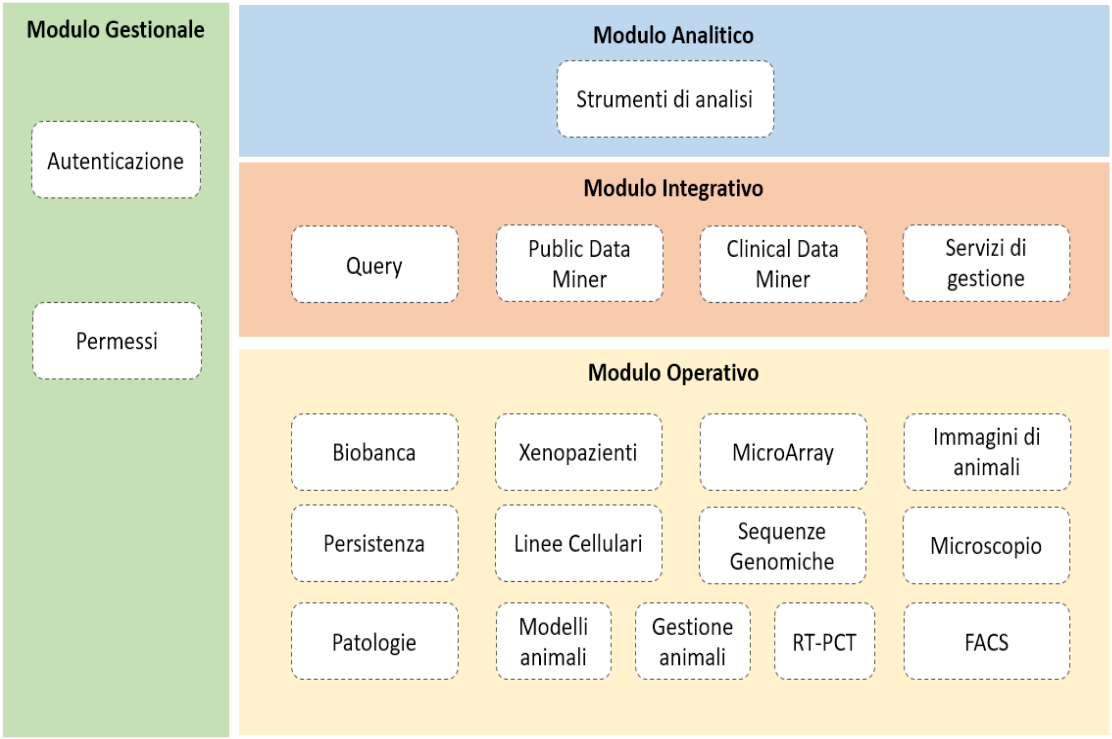

Figura 3.1: Livelli Architetturali LAS 1.0

I diversi moduli possono essere installati su macchine diverse, in modo da distribuire il carico di lavoro di tutto il sistema. Ogni modulo del LAS rappresenta un'applicazione web, realizzata utilizzando il paradigma di programmazione MVC (Model View Controller).
Il servizio offerto dal server è stato sviluppato a partire dal framework open-source Django: il linguaggio di programmazione lato server quindi è Python.
Utilizzando Django, per la precisione, il modello MVC si trasforma, come mostrato in Figura 3.2, in un modello MTV (Model-Template-View), i cui elementi principali sono:

- URL Dispatcher: viene comunemente chiamato Controller nel modello MTV e il suo compito è di selezionare l'opportuna View in base all'URL utilizzato dall'utente per effettuare la richiesta HTTP dal proprio dispositivo.

- View: si riferisce al Controller nel modello classico MVC e gestisce tutte le logiche di business. Modifica opportunamente il Model (modello dati) e seleziona il Template (interfaccia) da restituire all'utente.

- Model: il suo compito è quello di gestire la persistenza dei dati eseguendo su di essi le classiche operazioni CRUD. Per un modello possono esistere N rappresentazioni diverse (Template).

- Template: è l'equivalente della Vista nel modello MVC. Rappresenta l'interfaccia incapsulata in una risposta HTTP restituita all'utente sulla base di una determinata richiesta.

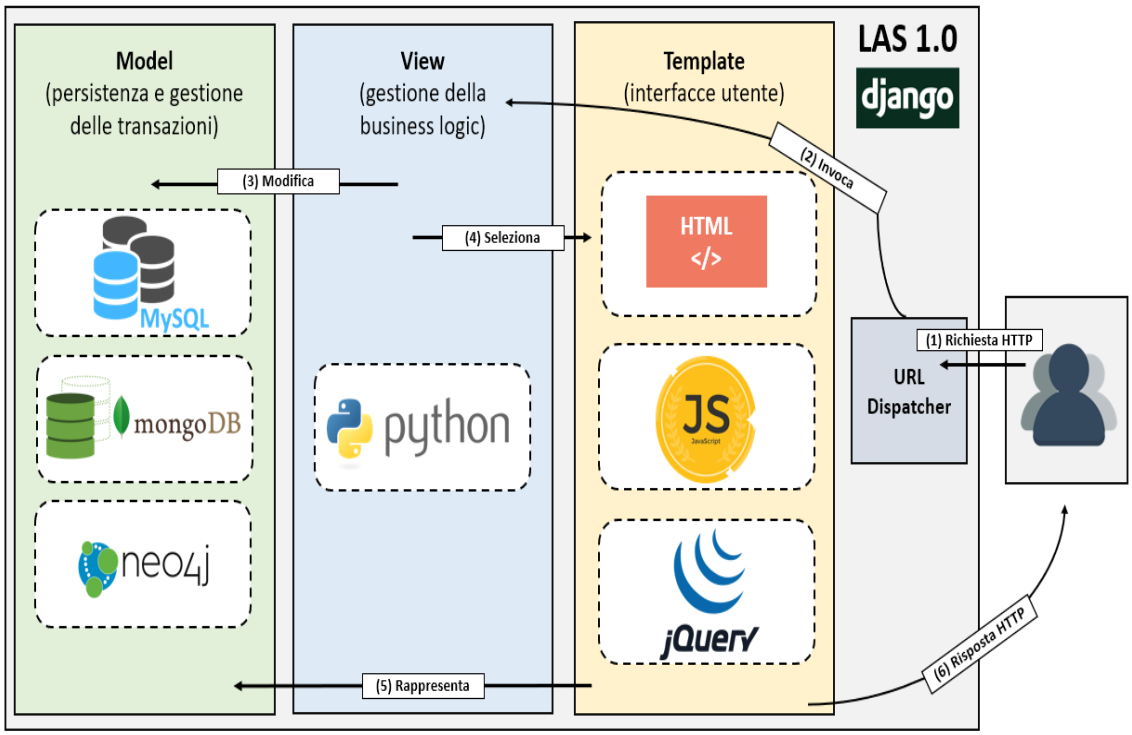

Figura 3.2: MTV in Django per il LAS 1.0

Dal punto di vista client è stata creata una MPA (Multiple Page Application) basata su tecnologie quali HTML, Javascript, jQuery, jQueryUI. Per l'utilizzo del sistema non vi sono requisiti stringenti e quindi basta utilizzare un qualsiasi dispositivo in grado di connettersi ad Internet.

I modelli di dato e le procedure integrati nella piattaforma cercano di conformarsi alle migliori pratiche e standard ampiamente adottati dalla comunità di ricerca in generale.
La gestione della persistenza dei dati raccolti durante il lavoro in laboratorio da parte degli operatori e delle annotazioni create durante il processo di analisi è possibile mediante l'impiego di un database relazionale, ovvero MySQL e due database non relazionali quali MongoDB e Neo4J.

3.2 Necessità di innovazione

Attualmente le applicazioni web a pagina singola hanno creato una nuova base per lo sviluppo di applicativi moderni e reattivi. Queste permettono di spostare parte della logica operativa sul client alleggerendo il carico delle operazioni svolte lato server, le quali si limitano all'esposizione di un servizio REST in grado di interagire con i database di riferimento.
Alcune tecnologie per sviluppare applicativi web, utilizzate in un passato non troppo lontano, si sono dimostrate un fallimento dal punto di vista dell'efficienza e delle performance.

Un esempio, a tal proposito, è sicuramente quello di Google il quale ha abbandonato il suo primo framework AngularJS in favore di una nuova tecnologia che ha rivoluzionato le tecniche di programmazione classiche: Angular (2+).
Nello specifico del LAS 1.0 l'impiego di JQuery e di un modello MPA hanno rappresentato un forte limite per l'operatività del progetto.
La scelta delle soluzioni di modellazione della struttura dei dati, la selezione e l'integrazione dei plugin e la personalizzazione del codice sono tutti sotto il controllo (e la discrezione) dello sviluppatore. Di conseguenza, il codice risultante spesso non è riutilizzabile tranne che per l'implementazione di GUI molto simili. Inoltre, gli aggiornamenti del codice possono essere molto impegnativi, specialmente se eseguiti da diversi sviluppatori.

La gestione di tre database differenti (MySQL, MongoDB e Neo4J), come mostrato in Figura 3.3, ha aumentato la complessità del sistema in maniera non indifferente, rendendo spesso poco chiara e trasparente la memorizzazione dei dati per gli sviluppatori.
La necessità di modellare entità chiave in più database solleva problemi critici nel mantenimento della coerenza dei dati e introduce una certa quantità di ridondanza. Inoltre, quando è necessaria l'integrazione di informazioni composite relative alla stessa entità, il sistema deve solitamente interrogare più di un database per recuperare tutti i dati necessari.
Ad esempio, se abbiamo bisogno di recuperare gli attributi intrinsechi di un insieme di campioni insieme alla loro posizione, il sistema interroga sia la banca biologica che i database di archiviazione. Questo approccio inevitabilmente aggiunge una complessità legata alle interazioni necessarie e rallenta il sistema generale. Le procedure di scrittura dei dati sono affette da inconvenienti simili. Durante la maggior parte dei processi di scrittura, diverse richieste API vengono inviate a più moduli. Questa soluzione richiede una gestione distribuita dei rollback, introducendo ritardi dovuti alla latenza della rete.

Sempre in ambito database un altro aspetto critico è la complessità dei modelli sviluppati. I database relazionali non sono adatti per acquisire modelli di dati dinamici che si evolvono rapidamente. Ogni volta che un'entità acquisisce nuove funzionalità (attributi o relazioni), lo sforzo necessario per adattare il database e il relativo codice può essere enormemente grande. Naturalmente, ciò si traduce in costi molto elevati e un lungo periodo di produzione.

Per quanto riguarda il lato server, il servizio sviluppato non rispettava il modello di maturità proposto da Richardson e per questo il backend non poteva essere considerato un vero e proprio servizio REST [5]. Il modello era basato principalmente su due server: uno destinato alla gestione dei tre database e uno destinato alla logica applicativa e, quindi, alle interazioni con gli utenti. Il punto debole di questa architettura era il fatto di non essere orientata al micro-servizio e di avere una limitata scalabilità orizzontale.

Un altro aspetto critico del LAS 1.0 è il basso tasso di riutilizzo del software. Infatti, anche se le soluzioni software sono condivise tra gli sviluppatori, ogni programmatore tende ad adattare il codice ai requisiti delle funzionalità della GUI e al suo stile di programmazione. Per questo motivo, la percentuale di codice comune riutilizzato più di una volta è estremamente bassa.

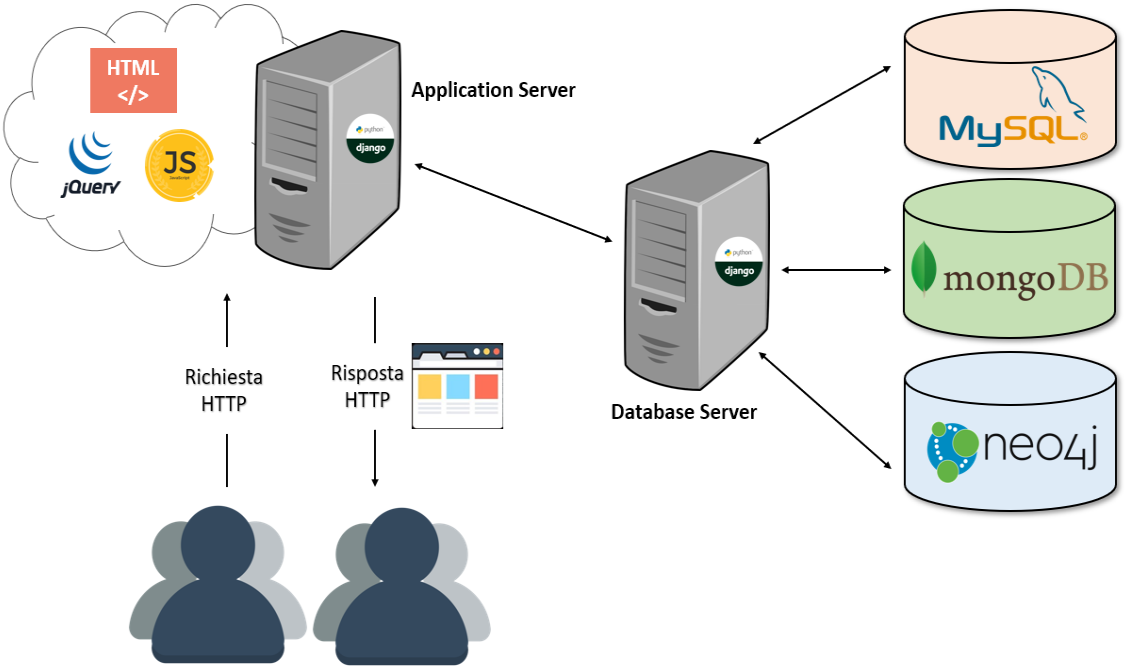

Figura 3.3: Architettura Generale Framework 1.0

L'analisi di alcuni report da parte degli utilizzatori finali ha permesso, inoltre, di riscontrare alcune criticità nel suo utilizzo.
Alcuni problemi sono associati alla perdita dei dati dovuti a malfunzionamenti durante lo svolgimento delle attività quotidiane di laboratorio: il framework è fortemente dipendente dall'utilizzo della rete e solo per alcune interfacce è previsto un meccanismo di recupero della sessione. Questo comporta che ogni qualvolta un operatore utilizza una interfaccia sprovvista del meccanismo di recovery e riscontra un errore nel sistema deve obbligatoriamente contattare, tramite l'utilizzo di una sezione ticket, lo staff di assistenza il quale dovrà ristabilire il corretto funzionamento del sistema per quel determinato utente analizzando opportuni file di log e ripristinando, quando possibile, la perdita dei dati.

Infine, per quanto riguarda il progetto della Biobanca, in questi anni ha rappresentato un semplice sotto-modulo all'interno del framework del LAS. Quello che si vuole realizzare con il nuovo progetto è l'accesso a un applicativo attivo su un dominio differente che permetta di usufruire del suo contenuto sia ad un pubblico esterno sia ai ricercatori (tramite quindi un collegamento tra i due applicativi).

3.3 Framework 2.0

Il nuovo progetto mira ad estendere la piattaforma attuale al fine di semplificare l'implementazione dei moduli dedicati alla gestione dei nuovi processi di laboratorio.
L'idea di base è quella di rinnovare completamente il LAS e renderlo parte integrante di una realtà ben più ampia composta da una Biobanca dati (accessibile sia da parte degli operatori del LAS per motivi di ricerca, sia da un pubblico che voglia registrarsi

e accedere alle analisi e agli studi esposti) e da un Blog informativo. L'intero sistema sarà inoltre creato in maniera tale da rendere possibile la sua distribuzione e la sua configurazione per altre istituzioni che condividono obiettivi comuni a quelli dell'istituto di Candiolo IRCCS.
L'obiettivo del LAS 2.0 è quello di armonizzare la gestione dei dati e delle procedure svolte all'interno dei laboratori, garantendo alte prestazioni e una semplicità di utilizzo in condizioni ostili.

Dal punto di vista del programmatore, la nuova architettura dovrebbe offrire alcuni vantaggi nella modellazione e nello sviluppo di nuovi tipi di dati e procedure. In particolare, l'obiettivo principale è ridurre i tempi di produzione. Per questo motivo, la nuova piattaforma è stata progettata con lo scopo di aumentare il riutilizzo del codice seguendo schemi di programmazione ben definiti e adottando anche un'architettura modulare sul lato client. Grazie alla modularità, gli sviluppatori saranno in grado di introdurre nuove funzionalità nelle GUI con uno sforzo minore, mentre il riutilizzo del codice ridurrà gli errori di programmazione

Un ulteriore obiettivo è definire descrizioni formali delle strutture dati al fine di standardizzare le comunicazioni tra i componenti del sistema, consentendo anche l'applicazione di validatori che garantiscano la coerenza dei dati. A tal proposito esiste la possibilità di adottare un unico database di tipo NoSQL, il quale può portare una maggiore flessibilità nella modellazione dei requisiti senza dover per forza definire uno schema iniziale. Per prevenire potenziali problemi di integrità dei dati, che potrebbero emergere come conseguenza dell'adozione di strutture di dati altamente flessibili, sia i modelli di dati che le loro interazioni potranno essere, eventualmente, guidati da una base di conoscenze basata sull'ontologia.

3.3.1 LAS 2.0

Il primo degli applicativi della nuova famiglia è il LAS 2.0. Come il suo predecessore il suo scopo principale è quello di supportare il lavoro svolto da parte degli operatori nei laboratori.

Si è posta particolare attenzione per la gestione di una rete di ricercatori che possa lavorare liberamente all'interno di progetti differenti, manipolando solo i dati per cui è autorizzato, con la possibilità di scambiare informazioni e contattare attraverso un supporto di messaggistica gli altri utenti. Tutto questo è possibile grazie alla definizione di alcune regole e alla gestione opportuna dei permessi.
In particolare possono essere definiti alcuni progetti di ricerca (ognuno con determinati obiettivi finali), ai quali aderiscono determinati gruppi di lavoro (Working Group). Ogni WG è formato da un Principal Investigator (PI), da alcuni Vice PI e da utenti semplici (Figura 3.4). La distinzione di questi tre ruoli ha uno scopo sia legale sia gestionale, in quanto il PI è il principale responsabile delle attività svolte da parte di uno o più gruppi di lavoro ognuno appartenente a un determinato progetto di ricerca. Il suo ruolo è quello di essere il principale responsabile del comportamento degli utenti del WG, dei campioni raccolti e delle analisi effettuate su di essi.

Per accedere a questo applicativo è necessario effettuare un'apposita registrazione (spiegata nel dettaglio nel Capitolo 6). Inoltre, un apposito modulo amministrativo

è stato previsto per la gestione sia del LAS sia della Biobanca dati (maggiori dettagli sono presenti nel Capitolo 8): in questa maniera la gestione degli utenti, dei loro permessi sui dati e delle loro funzionalità avviene in maniera centralizzata.

Rispetto al progetto precedente le innovazioni principali sono in termini di tecnologie e tipologia di architettura utilizzate, le quali garantiranno una maggior efficienza e semplicità d'uso durante il suo utilizzo. Le logiche di autenticazione e di gestione dei permessi sono state rivoluzionate, mentre i moduli operativi saranno rimodellati a livello grafico e funzionale per migliorare il generale flusso operativo e l'esperienza utente.

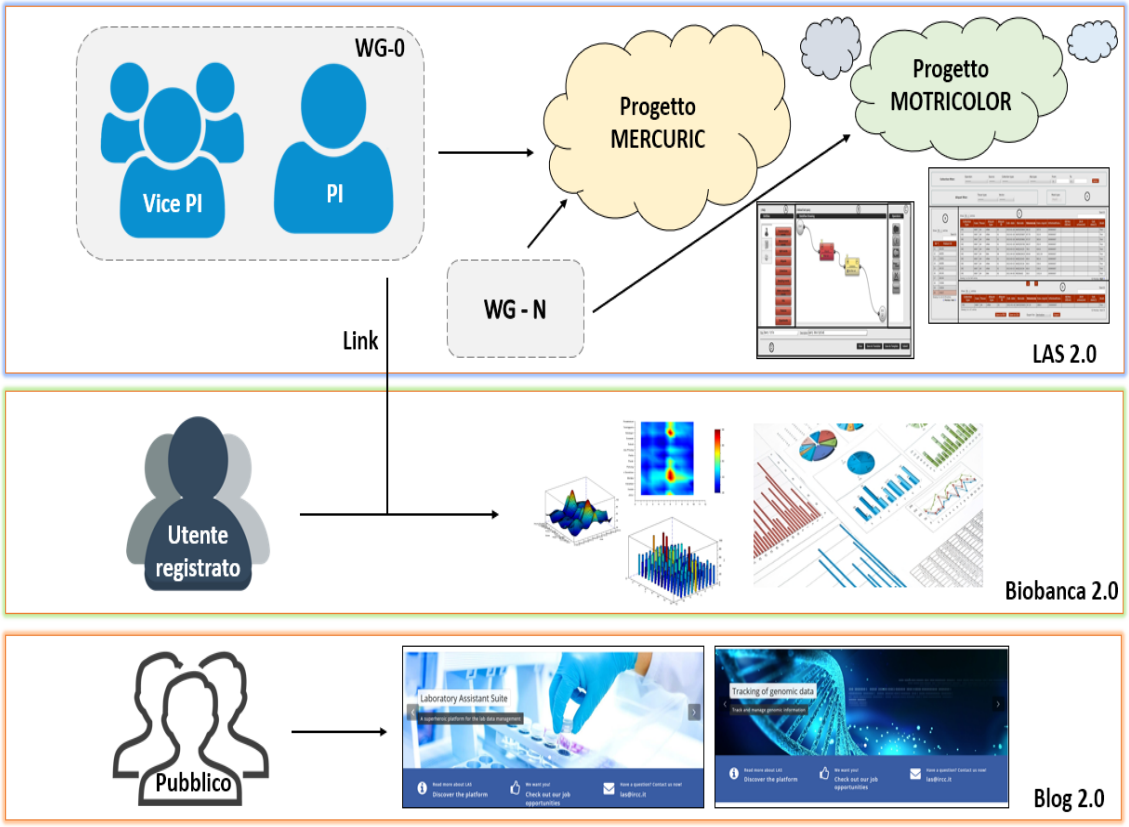

Figura 3.4: Applicativi del nuovo framework

3.3.2 Biobanca 2.0

Le Biobanche sono definite [6] come "unità di servizio, senza scopo di lucro diretto, finalizzate alla raccolta e alla conservazione di materiale biologico umano utilizzato per diagnosi, per studi sulla biodiversità e per ricerca". Le biobanche così intese nascono sul modello organizzativo dei Centri di Risorse Biologiche (CRB) definiti quest'ultimi dalla Organizzazione per la Cooperazione e lo Sviluppo Economico (OCSE) come centri che "forniscono servizi di conservazione di cellule viventi, di genomi di organismi e informazioni relative all'ereditarietà e alle funzioni dei sistemi biologici. Conservano banche di organismi coltivabili (microrganismi, cellule vegetali, animali e umane), parti replicabili di essi (genomi, plasmidi, virus, cDNA), organismi vitali ma non più coltivabili, cellule e tessuti, così come anche banche

dati contenenti informazioni molecolari, fisiologiche e strutturali rilevanti per quelle collezioni". In quest'ottica le biobanche trovano all'interno dei CRB la loro sede e il loro completamento naturale [6].

Le fonti più comuni di tessuti e organi umani per questa tipologia di banca sono:

- materiale derivato da interventi diagnostici o terapeutici;
- materiale specificamente donato per un progetto di ricerca e conservato per un successivo uso;
- materiale proveniente da persone decedute e sottoposte ad autopsia;
- materiale derivante da un trapianto ritenuto non utilizzabile.

I risultati della ricerca in ambito tumorale sono strettamente dipendenti dalla affidabilità e portata delle informazioni conservate insieme ai tessuti. Ne consegue l'importanza di sviluppare un completo applicativo per la corretta gestione delle informazioni raccolte durante tutti i processi operativi che possa essere ampiamente riconosciuto e utilizzabile dalla comunità scientifica.

Nel progetto precedente la Biobanca rappresentava un semplice sotto-modulo del livello operativo. Nel nuovo framework è stato sviluppato un secondo applicativo, su un dominio separato, interamente dedicato alle sue funzionalità. Il suo compito è quello di rappresentare un punto di accesso per un nuovo portale di ricerca e di richiesta campioni (con relative informazioni associate).

Per quanto riguarda la registrazione è stato previsto un modulo di autenticazione degli utenti appartenenti ad un pubblico esterno alla realtà istituzionale. L'applicazione risulta invece automaticamente disponibile attraverso un opportuno collegamento ad un utente registrato nel primo applicativo. Sostanzialmente, essere registrato per il LAS 2.0 permette di accedere in maniera diretta anche alla Biobanca dati, mentre un utente esterno che si sottoscrive all'applicativo della Biobanca non potrà accedere al contenuto del LAS 2.0, per ovvie ragioni.

3.3.3 Blog

L'attività di ricerca svolta mediante l'utilizzo delle tecnologie del LAS e della Biobanca presuppone la partecipazione attiva dei cittadini in qualità di donatori. Oltre a illustrare il progetto ai pazienti e a raccogliere il loro consenso per la donazione dei propri biomateriali, è necessario rendere il pubblico consapevole delle attività svolte nei laboratori e degli obiettivi della comunità scientifica.

Risulta utile dimostrare che il lavoro svolto sia di beneficio per la comunità a carattere generale, in maniera tale da sensibilizzare il cittadino ancor prima del paziente. Purtroppo per colpa della malattia, il paziente risulta essere spesso in uno stato psicologico alterato in quello che potrebbe essere uno dei momenti più delicati della propria esistenza. E' quindi stato ritenuto opportuno, per ragioni etiche e di trasparenza, dare la possibilità a tutta la comunità di conoscere le linee guida principali delle attività svolte.

Il Blog rappresenta un piccolo sito che può essere personalizzato e configurato dall'istituzione che lo gestisce. Il suo scopo è quello di fornire informazioni di base relative alle istituzioni coinvolte, al lavoro svolto e ai progressi compiuti nell'ambito della ricerca tramite articoli, post e pagine web.
Questo frontend ha ruolo prettamente divulgativo rispetto alle altre due applicazioni principali. Per questo motivo sono state adottate tecnologie caratterizzate dalla semplicità di utilizzo e da una buona adattabilità.

3.4 Vantaggi del nuovo Framework

Le principali innovazioni rispetto al vecchio framework sono:

- Nuove Tecnologie: è stato adottato un modello di tipo SPA (Single Page Application) il quale prevede l'esistenza di due grandi ecosistemi: un servizio REST (backend) e un applicativo client (frontend) suddiviso su tre domini differenti. Per il processo di rinnovamento è stata effettuata un'attenta e scrupolosa analisi dei framework e delle tecnologie disponibili, considerando pro e contro di ognuno di essi e valutando le potenzialità e le conoscenze in possesso da parte del team di sviluppo.

- Modularità: questo aspetto potrebbe non risultare nuovo, certamente dal punto di vista server l'organizzazione del servizio REST richiede una certa modularità ma questa era completamente assente nel progetto iniziale per quanto riguarda l'applicativo client.
  La SPA sviluppata è stata organizzata in opportuni moduli. Il modulo principale viene caricato durante il primo avvio dell'applicazione mentre tutti gli altri moduli vengono caricati con la tecnica del lazy-loading, ovvero solamente su richiesta. Questo permette di avere un client reattivo e fluido, con un'ottima struttura di progetto facilmente modificabile e mantenibile.

- Interfacce: dal punto di vista dell'utente, la piattaforma potrà eventualmente introdurre un meccanismo di personalizzazione delle interfacce in base alle sue preferenze e alle pipeline sperimentali definite in ciascun laboratorio. Inoltre, le interfacce potranno essere progettate in modo da tenere conto di principi consolidati, al fine di fornire un'esperienza utente preziosa.

- Progettazione di componenti: la progettazione delle interfacce client si basa sul concetto di componente. E' possibile progettare un componente in maniera generale, immaginiamo un grafico utilizzato per mostrare alcune analisi sui dati o delle Smart Table, per poterlo riutilizzare in diversi moduli applicativi. Questo concetto permette un ottimo riutilizzo del codice riducendo i tempi di sviluppo.

- Integrazione dati: la piattaforma fornisce una serie di strumenti per integrare facilmente dati eterogenei, analizzare i dati con software statistico e basato su domini, monitorare tutte le attività e pianificare pipeline di lavoro complesse.

- Paginazione dei dati: questa attività, del tutto assente nel framework 1.0, prevede la suddivisione di una grossa mole di dati all'interno di più pagine in sequenza. Per fare un esempio pratico, immaginiamo di avere una tabella che mostra il contenuto di una sezione della biobanca. Se i dati relativi a quella sezione avessero dimensioni eccessive, durante la risposta dal server al client, verrebbero trasferite molte informazioni, sovraccaricando la rete o il rendering dell'applicativo la cui logica sarebbe obbligata a memorizzare ognuno di quei dati in locale.
  Utilizzando questo nuovo meccanismo quello che avviene, sostanzialmente, è la creazione di un proxy, il quale permette di visualizzare un sotto-insieme parziale dei dati effettivamente disponibili per evitare di sovraccaricare l'applicativo. Ogniqualvolta un utente necessità di una nuova porzione di dati, questa gli viene restituita su richiesta (tramite una opportuna logica a livello di interfaccia).

- Gestione dello stato: sempre a livello client è possibile mantenere uno stato sempre aggiornato durante le operazioni effettuate dal singolo utente (modello di programmazione Redux). Questo stato viene organizzato in un albero di sotto-stati i grado di evolvere nel tempo e salvato localmente attraverso le tecnologie di localstorage e indexedDB. Quest'ultimo, in particolare, permette di effettuare interrogazioni e ricerche basandosi sul meccanismo di indicizzazione.

- Cache dei dati: sia per quanto riguarda lo stato sia per le operazioni di modifica o aggiornamento dei dati relativamente ai componenti sviluppati nel client, è possibile effettuare un salvataggio intermedio dei dati nella cache del sistema prima di effettuare una richiesta definitiva al servizio REST. Il suo impiego può aiutare a sviluppare un servizio di recupero locale dei dati in caso di un errore imprevisto. Il suo impiego è utile per garantire un funzionamento base dell'applicazione anche in situazioni ostili come in assenza di rete.

- Applicativo della biobanca: oltre al frontend sviluppato per la gestione delle funzionalità operative del LAS 2.0 è stato previsto un applicativo relativo alla Biobanca il quale consente la gestione e visualizzazione dei dati raccolti ai fini di fornire informazioni per la ricerca. Questi applicativi vivono in domini separati.

- Supporto alla messaggistica: è previsto un sistema utilizzabile dai ricercatori per contattare altri membri del LAS e notificare i risultati ottenuti tramite messaggi (la sua descrizione è rimandata nel Capitolo 8).

- Distribuzione: il progetto è stato sviluppato con l'idea di poterlo distribuire in un prossimo futuro in maniera efficace e trasparente in modo tale da renderlo disponibile ad altre associazioni che richiedano una tecnologia simile nelle funzionalità ma in parte personalizzabile e configurabile per quanto riguarda l'aspetto e la gestione.

- Documentazione: durante la creazione di progetti complessi e articolati è indispensabile creare una documentazione di base da fornire agli sviluppatori. I team di sviluppo spesso hanno un certo ricambio di personale durante la

creazione di un progetto a lungo termine e, una volta messo in produzione, bisogna continuamente provvedere ad attività di manutenzione e aggiornamento del codice.

Capitolo 4

Tecnologie a confronto

In questo Capitolo è possibile accedere ad un confronto diretto con le maggiori tecnologie utilizzabili nello sviluppo di un progetto basato sulla creazione di un applicativo web per la gestione di una biobanca dati. Per ognuna di esse sono forniti una breve descrizione della tecnologia, vantaggi e svantaggi derivanti dal suo utilizzo.
Le scelte effettuate sono state influenzate dall'esistenza di un applicativo sviluppato in passato (il framework 1.0), il quale ha richiesto per certi aspetti una reingegnerizzazione e per certi aspetti una totale innovazione, utilizzando alcuni moderni strumenti.

4.1 Web Server

Per la gestione del web server esistono due principali soluzioni:

- Apache: [7] è un web server open source sviluppato dall'Apache Software Foundation. In generale l'applicativo ha il compito principale di ascoltare le richieste HTTP sulla porta 80 e consegna i documenti in formato HTML per la visualizzazione. Vanta una struttura modulare e permette l'integrazione con altre tecnologie quali database, linguaggi di scripting lato server con supporto quindi alla gestione di pagine web dinamiche (scritte per esempio usando PHP e Python). Oltre al ruolo di web server può assumere quello di proxy server. Gestisce contenuti web sia di tipo statico sia di tipo dinamico.
- NGINX: [8] rilasciato nel 2004 fu ideato dallo sviluppatore russo Igor Sysoev. Gestisce esclusivamente contenuti web statici e la sua struttura risulta, come per il precedente, modulare e integrabile. In particolare le sue funzioni principali sono:
  - accelerazione delle applicazioni: permette una trasmissione più veloce del contenuto;
  - bilanciamento del carico: alleggerisce il server principale, distribuendo le richieste tra i vari nodi;
  - crittografia TLS: permette un trasferimento dati a canale sicuro;

- proxy: è utilizzabile come reverse o email proxy in base alle necessità architetturali;
- video streaming: offre alte prestazioni nello streaming;

Apache ogni volta che riceve una richiesta proveniente da un client crea un nuovo thread o un nuovo processo. Questo approccio ha rappresentato in passato un buona soluzione ma attualmente rappresenta un "collo di bottiglia" per le prestazioni. Per quanto riguarda i processi è necessario utilizzare tempo di CPU e allocare memoria separata mentre per il multithreading occorre una notevole capacità di calcolo per gestire il context switching: il processo attraverso il quale un sistema commuta da un processo o da un thread a un altro durante il quale devono essere caricati e salvati il registro CPU, le diverse tabelle e liste.
Dalla versione 2.4 è stato introdotto un meccanismo basato su eventi che prevede un listener thread il cui compito è di salvare le connessioni in entrata e terminare le connessioni non più necessarie (anche connessioni keep alive) riducendo in questo modo l'uso delle risorse. Tuttavia rimane ancora il forte limite derivante dal context switching che si verifica durante e dopo il passaggio di consegna dal thread listener ai thread worker.

Contrariamente, il server NGINX, utilizza un meccanismo che si basa su eventi che non richiede di creare un nuovo thread o processo per ogni connessione in entrata. Attraverso l'Event Loop le richieste vengono così elaborate in maniera asincrona all'interno di un unico thread e, in questo modo, vengono risparmiati memoria locale e tempo migliorando in generale le prestazioni del web server.

Sulla base delle considerazioni effettuate e vista l'integrabilità con una struttura basata sui concetti di cloud e micro-servizi è stato scelto NGINX come tecnologia base per la gestione del web server.

4.2 Virtualizzazione

La creazione di sistemi distribuiti richiede la progettazione e messa in campo di un certo numero di componenti. Quando si ha a disposizione determinati elementi hardware e software è necessario ottimizzarne il loro impiego attraverso l'utilizzo della tecnica della virtualizzazione.
Questo si traduce in un risparmio in termini di risorse utilizzate, spazio fisico impiegato, tecnologie e costi di manutenzione.
La virtualizzazione riduce in generale la necessità di investire capitale in nuovo hardware. Permette di ridistribuire velocemente i server non appena si rendono disponibili: questo approccio è utilissimo in un ambiente orientato ai micro-servizi.
Attraverso la virtualizzazione è possibile fornire un supporto all'automazione, la quale consente di avviare una VM (Virtual Machine) in poco tempo e allocare nuovi carichi di lavoro, permettendo così la scalabilità orizzontale del servizio offerto.
Questo approccio offre caratteristiche quali la velocità, la flessibilità e ottime prestazioni richieste dalle odierne aziende altamente connesse al web.
Principalmente è possibile utilizzare due approcci:

1. Virtual Machine: consente l'ottimizzazione delle risorse hardware disponibili, ripartendole tra diverse configurazioni software indipendenti tra loro. Ogni VM occupa spazio fisico per il sistema operativo ed è in grado di ospitare quindi uno o più applicativi.
   Si basa sul meccanismo dell'Hypervisor, uno standard de facto del mondo enterprise per la virtualizzazione di server applicativi basati sul principio della virtualizzazione delle risorse hardware. L'Hypervisor può controllare ed interrompere eventuali attività pericolose, fornisce quindi una molteplicità di livello di protezione della CPU. Può allocare, inoltre, le risorse dinamicamente quando e dove necessario, riducendo così il tempo necessario alla messa in opera di nuovi sistemi, isolare l'architettura nel suo complesso da problemi a livello di sistema operativo ed applicativo, abilitare una gestione più semplice di risorse eterogenee e facilitarne il collaudo e il debugging. Questo approccio era usato in passato quando il costo delle macchine fisiche era molto più grande rispetto al costo degli applicativi sviluppati.

   - Vantaggi: nonostante le macchine virtuali possano essere eseguite sulla stessa macchina fisica, rimangono separate l'una dall'altra a livello logico (principio di isolamento).
     Qualora una VM sperimentasse un errore, sia per colpa di un arresto anomalo sia per motivi di sicurezza, questo non verrebbe esteso ad altre VM sulla stessa macchina ospitante o su altre macchine collegate.
     Inoltre, è possibile effettuare la migrazione delle VM sviluppate tra server virtuali locali o remoti in maniera decisamente molto più trasparente e semplice rispetto alle applicazioni tradizionali legate alla tipologia di hardware fisico (principio di mobilità).
   - Svantaggi: dato che ogni VM deve necessariamente avere un sistema operativo, molto dello spazio hardware destinato alla virtualizzazione viene utilizzato per l'allocazione di questo e non viene ottimizzato. Infine, bisogna sottolineare che l'Hypervisor utilizza due scheduling della memoria, uno previsto da parte del sistema operativo della macchina fisica ospitante e uno previsto da parte della VM ospitata.

2. Container: si basa sul concetto di virtualizzazione software utilizzando un unico sistema operativo Linux, che dalla versione 2.4 ha introdotto la "system call clone" (la quale ha permesso lo sviluppo di questa nuova tecnica).
   La virtualizzazione viene organizzata in contenitori: i processi all'interno dello stesso contenitore possono comunicare tra loro perché vivono nello stesso ambiente condividendo lo stesso spazio dei nomi (namespace), mentre rimangono isolati da processi che appartengono a contenitori differenti sulla stessa macchina fisica.

   - Vantaggi: questa tecnica, come mostrato in Figura 4.1, permette di utilizzare un unico sistema operativo con un unico scheduling. Questo porta a risparmiare risorse hardware allocando decisamente meno spazio per ogni contenitore e per ogni applicativo.
     Inoltre, la tecnologia dei container permette di semplificare il deployment di qualsiasi applicazione senza doversi preoccupare della configurazione

dell'ambiente di esecuzione.
L'intero sistema è molto più leggero e rapido rispetto all'approccio con Hypervisor in quanto si basa su un unico sistema operativo.
La possibilità di pacchettizzare, non solo un applicativo ma anche un singolo componente, permette di avere un controllo più granulare orientato al micro-servizio.
Infine, è da sottolineare come questo approccio possa permettere la risoluzione del problema delle librerie condivise: il file system di ogni contenitore è indipendente da quello degli altri contenitori e questo permette quindi la possibilità di poter disporre di diverse versioni della stessa libreria senza che possa esserci un contrasto tra loro.

- Svantaggi: la difficoltà di gestione quando il numero di container aumenta è molto elevato. Un altro punto debole è proprio la condivisione dello stesso kernel del sistema operativo. A livello teorico, uno dei container potrebbe compromettere la stabilità del kernel stesso, influenzando anche gli altri.

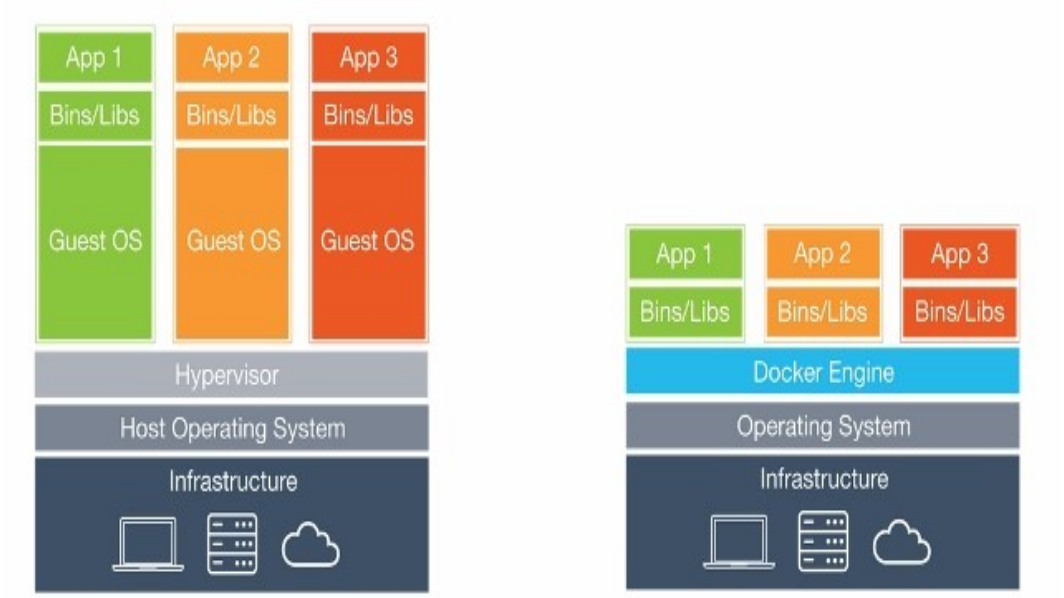

Figura 4.1: Virtualizzazione: Hypervisor vs Container

Utilizzare un approccio basato sulla virtualizzazione software tramite contenitori permette di avere un miglior impiego delle risorse. In questo ambito sono stati valutati due potenti strumenti:

- Kubernetes: [9] è una tecnologia open source per la gestione dei contenitori sviluppato dal team di Google. Permette di eliminare molti dei processi manuali coinvolti nella distribuzione delle applicazioni e di gestire in maniera semplice ed efficiente cluster di host su cui vengono eseguite. Uno dei vantaggi principali derivante dal suo utilizzo è quello utilizzare le ricerche fatte da Google, sfruttando quindi l'esperienza acquisita con gli anni.
Possiede una comunità competente e in crescita la quale mette a disposizione

risorse utili.
Può essere eseguito su un servizio cloud pubblico o in locale e risulta facile da apprendere e implementare.

- Docker: [10] come piattaforma ha introdotto molti vantaggi nella creazione e nell'esecuzione di applicazioni. Tramite questa tecnologia, un'applicazione può essere costruita, distribuita e ridimensionata rapidamente. Inoltre, può essere eseguita sempre e ovunque nonostante le risorse di sistema siano minime. Particolare attenzione è da porre alla modalità Swarm:

  > La modalità Swarm è una funzione di Docker che fornisce funzionalità di orchestrazione dei contenitori predefinite, inclusi il clustering nativo di host Docker e la pianificazione dei carichi di lavoro dei contenitori. Un gruppo di host Docker forma un cluster "swarm" quando i relativi motori Docker vengono eseguiti insieme nella "modalità swarm". Uno swarm è costituito da due tipi di host contenitore: nodi di gestione e nodi del ruolo di lavoro (mostrato in Figura 4.2). Ogni swarm viene inizializzato mediante un nodo di gestione e tutti i comandi dell'interfaccia della riga di comando di Docker per il controllo e il monitoraggio di uno swarm devono essere eseguiti da uno dei relativi nodi di gestione. I nodi di gestione possono essere considerati come dei "guardiani" dello stato Swarm; insieme formano un gruppo di consenso che mantiene il riconoscimento dello stato dei servizi in esecuzione nello swarm e il relativo compito è quello di garantire che lo stato effettivo dello swarm corrisponda sempre a quello previsto che viene definito dallo sviluppatore o dall'amministratore. I nodi del ruolo di lavoro vengono orchestrati dallo swarm di Docker tramite nodi di gestione. Per aggiungere uno swarm, un nodo del ruolo di lavoro deve utilizzare un "token di aggiunta" che è stato generato dal nodo di gestione quando lo swarm è stato inizializzato. I nodi del ruolo di lavoro ricevono ed eseguono semplicemente le attività dei nodi di gestione, pertanto non richiedono (e non dispongono) di alcun riconoscimento dello stato dello swarm. [11]

  Docker Swarm ha il vantaggio di essere strettamente integrato nell'ecosistema Docker e utilizza le proprie API. Il suo sistema di filtraggio e schedulazione consente la selezione di nodi ottimali in un cluster per la distribuzione di container.
  Poiché è sviluppato da Docker stesso, Docker Swarm rimuove molte incompatibilità e altre differenze e si integra perfettamente.

Volendo analizzare vantaggi e svantaggi di entrambe le tecnologie, si può evincere che Kubernetes è facile da utilizzare e personalizzare. Questa soluzione è adatta per configurazioni particolari e mirate in ambienti molto complessi, mentre Docker è la soluzione migliore per i team che cercano di configurare un'applicazione in tempi brevi e caratterizzate da una bassa complessità.

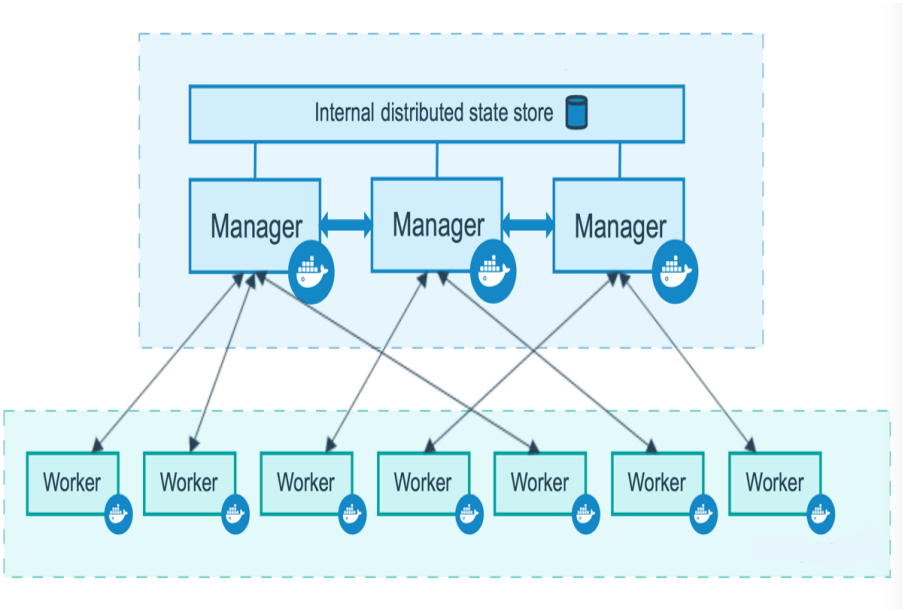

Figura 4.2: Dokcer Swarm

Uno dei limiti da considerare durante l'utilizzo di Docker Swarm è relativo alla comunità e al supporto, i quali si presentano meno espansivi e convenienti rispetto a Kubernates.
Quest'ultimo non rappresenta una soluzione completa e richiede plug-in personalizzati che a loro volta richiedono una configurazione: questo implica l'acquisizione di opportune competenze da parte degli sviluppatori in questo ambito.
Utilizzando Docker Swarm, invece, tutte le dipendenze sono gestite all'interno dell'ambiente nativo di Docker, rendendo l'installazione e la successiva configurazione davvero fluide e minimali.
Sulla base di queste considerazioni è stata scelta la tecnologia di virtualizzazione Docker.

4.3 Modello Dati

La scelta del modello dati è una delle scelte più importanti durante la fase di sviluppo di un progetto e, in particolare, rappresenta la prima da prendere in considerazione e modellare. Le scelte possibili ricadono su un modello relazionale o su un modello NoSQL:

- Modello relazionale: si basa sullo standard SQL (Structured Query Language) e generalmente garantisce le quattro proprietà ACID fondamentali (Atomicità, Consistenza, Isolamento e Durabilità), tuttavia la progettazione di uno schema iniziale dal quale l'intero sistema dipenderà porta ad avere poca flessibilità durante la fase di sviluppo e può comportare un dispendio eccessivo di risorse in caso di modifiche.
  Alcune operazioni di ricerca e di join talvolta diventano onerose e, inoltre, le strutture SQL sono fortemente incompatibili con strutture dati orientati agli

oggetti: basti immaginare che una riga di un database relazionale non può essere polimorfica mentre un oggetto si. Viceversa, le relazioni tra entità sono navigabili in diverse direzioni mentre le strutture ad oggetto devono mantenere in memoria un puntatore, il quale è unidirezionale.
Un'altra forte limitazione è rappresentata dalla difficoltà a scalare orizzontalmente: è difficile garantire le proprietà ACID in un sistema distribuito orientato ai micro-servizi.

- Modello NoSQL: in questo caso non è necessario definire subito uno schema, quindi l'intero sistema risulta molto più flessibile e disponibile al cambiamento, per questo si parla di proprietà di elasticità.
  Un altro punto forza di questa scelta è rappresentato dal supporto allo "sharding" in maniera del tutto trasparente alle applicazioni: suddividere il database in blocchi più piccoli denominati "shard" e spargerli su un numero di server distribuiti.
  Questo modello garantisce quindi una buona scalabilità orizzontale e inoltre assicura alte prestazioni in lettura e scrittura del dato.
  Quando si sviluppa un sistema distribuito tuttavia bisogna tener conto del teorema CAP (Consistency, Availability and Partition tollerance, Figura 4.3): è impossibile garantire contemporaneamente le proprietà di coerenza o consistenza, di disponibilità di servizio e di tolleranza di partizione, al massimo solo due di queste possono essere garantite.

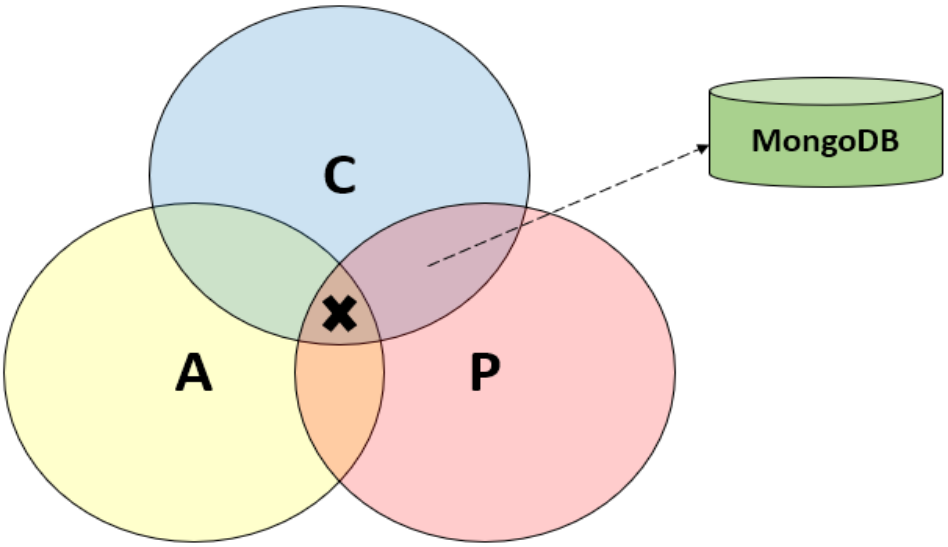

Figura 4.3: Teorema CAP

I database di tipo NoSQL si basano sulle proprietà BASE:

- Basic Available: indica la capacità di un sistema di garantire la disponibilità di servizio sulla base del Teorema CAP;
- Soft State: indica che lo stato del sistema può cambiare nel tempo anche in assenza di input (questo è dovuto al modello di convergenza eventuale);

- Eventually Consistent: il sistema diventerà consistente in una certa finestra temporale.

4.3.1 Database

Nel framework 1.0 venivano adottati MySQL [12] come database di tipo relazionale e MongoDB [13] in combinazione con Neo4J [14] come database di tipo NoSQL.
Sulla base delle considerazioni effettuate in precedenza, il modello relazionale è stato abbandonato in favore di una gestione completa da parte di un unico database della tipologia NoSQL.

MongoDB è un database non relazionale, orientato ai documenti. La sua struttura si basa su documenti in stile JSON con schema dinamico (comunemente chiamato formato BSON), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce.

Neo4J, invece, è un database open source orientato ai grafi. La sua caratteristica principale è di essere totalmente transazionale e solitamente viene integrato nelle applicazioni permettendone il funzionamento stand alone.
Gli elementi principali di questa tipologia di database NoSQL sono:

- Nodi: sono caratterizzati da proprietà e sono collegati ad altri nodi mediante relazioni. Ogni nodo può avere una o più etichette che ne descrivono il ruolo che assume nel grafo.
- Relazioni: rappresentano un collegamento direzionale tra due nodi (un nodo sorgente e un nodo destinazione). Le relazioni possono essere multiple per uno stesso nodo e anche ricorsive.
- Proprietà: sono coppie chiave-valore associate ad un nodo, possono essere indicizzate e vincolate.
- Etichette: sono informazioni associate ad un nodo. Servono per raggruppare insiemi di nodi con caratteristiche comuni. Possono essere indicizzate per velocizzare la ricerca dei nodi nel grafo.

All'interno di una transazione è possibile creare nodi e assegnar loro delle proprietà. Il suo utilizzo è particolarmente consigliato per la classificazione dei dati biologici, e quindi si adatta perfettamente alla destinazione d'uso di questo progetto.
Neo4J può essere utilizzato perfettamente in combinazione con MongoDB e per questo motivo nel framework 1.0 ne era stato previsto l'utilizzo:

> "Gli sviluppatori di MongoDB hanno fornito il progetto mongo-connector che fornisce un meccanismo per l'ascolto di tutte le operazioni di aggiornamento in MongoDB e facilita l'operazione di mirroring di tali aggiornamenti su un altro sistema. Dalla documentazione del connettore mongo: mongo-connector crea una pipeline da un cluster MongoDB a uno o più sistemi di destinazione, come Solr, Elasticsearch o un altro cluster MongoDB. Sincronizza i dati in MongoDB verso il target, poi passa all'oplog

di MongoDB, mantenendo le operazioni in MongoDB in tempo reale. È stato testato con Python in versione 2.6, 2.7, 3.3 e 3.4. Per facilitare la sincronizzazione dei dati da MongoDB a un'istanza Neo4j, la comunità ha implementato un Doc Manager Neo4j per mongo-connector. È inteso per la sincronizzazione unidirezionale da MongoDB a Neo4j, in cui entrambi i database sono in esecuzione per sfruttare i punti di forza di ciascun database nell'applicazione." [15]

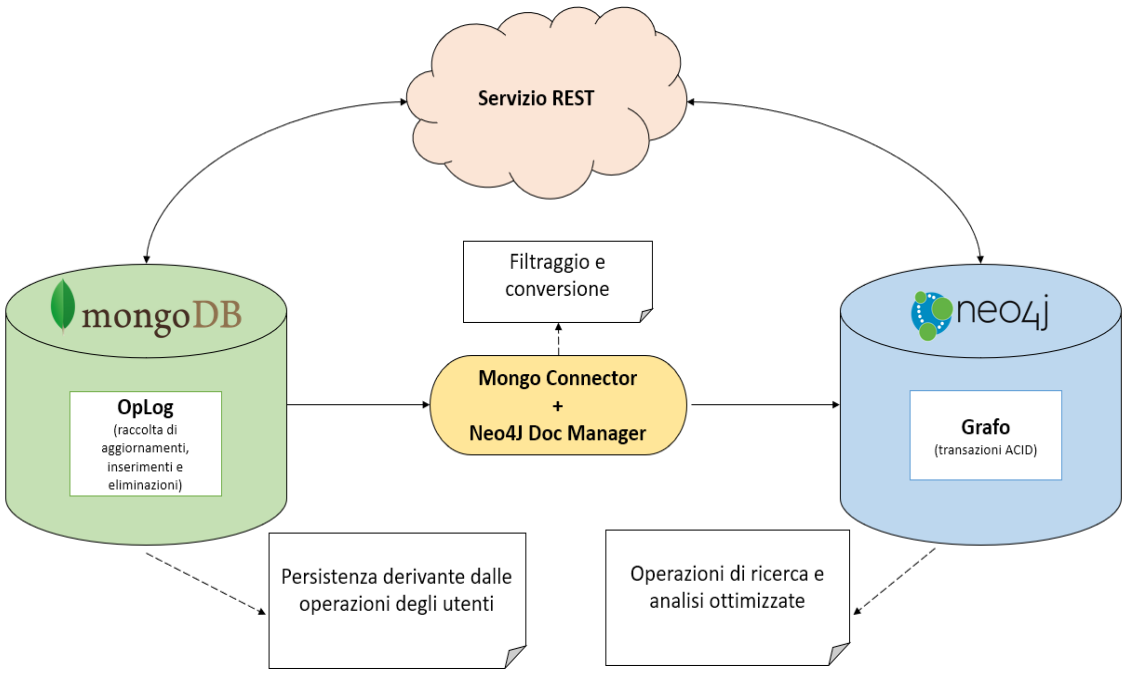

Figura 4.4: MongoDB Connector con Neo4J Doc Manager

La struttura a grafo di Neo4j risulta essere efficiente per la gestione di strutture ad albero estratte da file system, reti e file XML. L'esplorazione di queste strutture risulta decisamente più veloce e immediata rispetto alle tabelle del modello relazionale: ogni nodo è associato all'indice delle proprie relazioni in uscita e in ingresso, in questo modo la velocità di attraversamento del grafo non risente delle dimensioni totali del database ma solo della densità dei nodi attraversati durante l'interrogazione.
Per questo motivo, come mostrato in Figura 4.4, può risultare utile adottare una soluzione simile in determinati casi d'uso per poter trasferire parte del contenuto di MongoDB verso Neo4J per eseguire un'ottimizzazione delle query sui dati biologici memorizzati e beneficiare di un supporto alle transazioni.

Nel framework 1.0 del progetto Neo4J era impiegato principalmente per tenere traccia dei rapporti gerarchici tra le entità, delle relazioni di possesso tra entità e working group e dei loro meccanismi di ereditarietà, inoltre era utilizzato per rappresentare le interdipendenze tra concetti genomici.
Neo4J rappresentava un ottimo supporto per le transazioni di tipo ACID rispetto a MongoDB (prima della versione 4.0).

Grazie alla pubblicazione della nuova versione di MongoDB 4.0 risulta ottimale l'utilizzo di un'unica tecnologia per quanto riguarda i database e la persistenza.

Una delle novità è quella di fornire un supporto per le transazioni ACID rendendo questo database ancora più competitivo sul mercato di quanto non lo fosse già.

4.4 Framework Backend

Per quanto riguarda il framework destinato allo sviluppo lato server due scelte ottimali sono rappresentate da:

- Spring: [16] il suo scopo è quello di gestire la complessità legata allo sviluppo di applicazioni in ambito enterprise.
  Si basa su tre concetti principali:

  1. IoC (Inversion of Control): chiamato comunemente "Principio di Hollywood", è un modello di programmazione che realizza l'accoppiamento tra gli oggetti in fase di esecuzione, invece che in fase di compilazione come nella programmazione tradizionale.
  2. DC (Dependency Injection): tutte le dipendenze vengono create da un contenitore (Spring Container) e successivamente vengono iniettate nel programma principale sotto forma di proprietà negli oggetti. Questo processo si basa sull'esistenza di oggetti Bean il cui ciclo di vita è gestito interamente dal contenitore (mostrato in Figura 4.5).
  3. AOP (Aspect Oriented Programming): è un paradigma di programmazione in grado di descrivere i comportamenti trasversali dell'applicazione separandoli dal dominio applicativo. Si possono definire funzionalità comuni a più oggetti in un unico posto, chiamati "comportamenti trasversali", evitando di ripetere la logica comune e applicandola a più moduli.

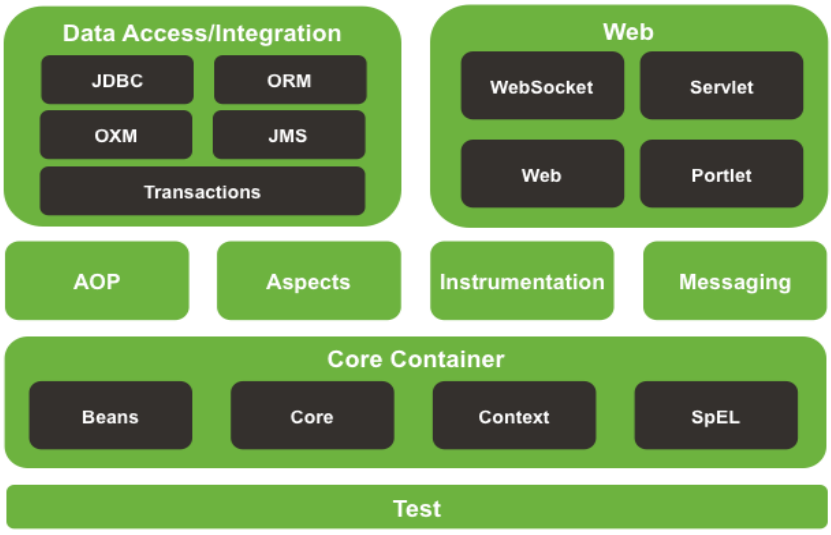

Figura 4.5: Spring Framework

Spring è inoltre un framework leggero caratterizzato da una struttura estremamente modulare, la quale permette di non stravolgere completamente l'architettura del progetto in caso di aggiornamento o modifica.
Prevede, inoltre, accesso ad un insieme completo di strumenti per la gestione della complessità dello sviluppo software, fornendo un approccio semplificato ai più comuni problemi di sviluppo (accesso ai database, gestione delle dipendenze, testing, etc.).
Questa tecnologia si basa principalmente sull'utilizzo del linguaggio Java e le sue configurazioni sono possibili sia tramite XML sia tramite l'uso di annotazioni.

- Django: [17] fornisce diverse funzionalità destinate a facilitare lo sviluppo di applicazioni per la gestione di contenuti Web. E' possibile usufruire di nuove funzionalità installandone i plug-in di riferimento e gode di un insieme di robuste API per la gestione del database.
  La sicurezza è una pietra miliare per questo framework il quale cerca di aiutare gli sviluppatori a evitare molti errori comuni. Tra le sue caratteristiche principali troviamo la possibilità di scalare rapidamente e in modo flessibile.
  Questa tecnologia si basa sull'utilizzo del linguaggio Python.

Entrambe le tecnologie sono open source e godono del supporto di un'ottima comunità di sviluppatori. La scelta finale è ricaduta sull'utilizzo di Django in quanto il progetto da reingegnerizzare era già stato sviluppato precedentemente con questo framework. Questo ha permesso in parte il riutilizzo del codice lato server usato per l'implementazione di determinati servizi e di evitare la dilatazione dei tempi di apprendimento e aggiornamento necessari a un corretto utilizzo della tecnologia da parte del team di sviluppo.

4.5 Framework Frontend

Prima di distribuire un'applicazione Web, è necessario considerare l'obiettivo.
Le applicazioni Web stanno progressivamente sostituendo le vecchie applicazioni di tipo desktop. Sono più convenienti dal punto di vista dell'utilizzo, sono facili da aggiornare e non vengono associate a un dispositivo (non ci sono limiti in termini della tipologia di hardware e sistema operativo utilizzati). In questo ambito ci sono due principali modelli di design per le applicazioni web:

1. Applicazioni multi-pagina (MPA) [18]: inizialmente erano applicazioni in cui ogni cambiamento richiedeva di ricaricare completamente il contenuto della pagina da parte del server. Con l'introduzione di AJAX è stato introdotto uno scambio e aggiornamento del contenuto delle pagine in background effettuato in maniera asincrona. Questa soluzione migliora e consente di aggiornare solo alcune parti specifiche dell'applicazione. D'altra parte, aggiunge più complessità ed è più difficile da sviluppare rispetto a un'applicazione a pagina singola. Il server gestisce interamente il rendering della pagina web.

- Vantaggi: è un approccio solido, che permette una gestione centralizzata del contenuto. Permette una corretta gestione del SEO (Search Engine Optimization) e offre maggiori possibilità di classificare parole chiave in quanto esiste un contesto multi-pagina.
- Svantaggi: lo sviluppo del frontend e del backend sono strettamente accoppiati. La complessità aumenta e i tempi sono dilatati. Gli sviluppatori devono in generale utilizzare un unico framework per due compiti che potrebbero essere parallelizzati. La fluidità e l'efficienza dell'applicativo sono limitati da questo modello di design e dalle tecnologie che utilizza.

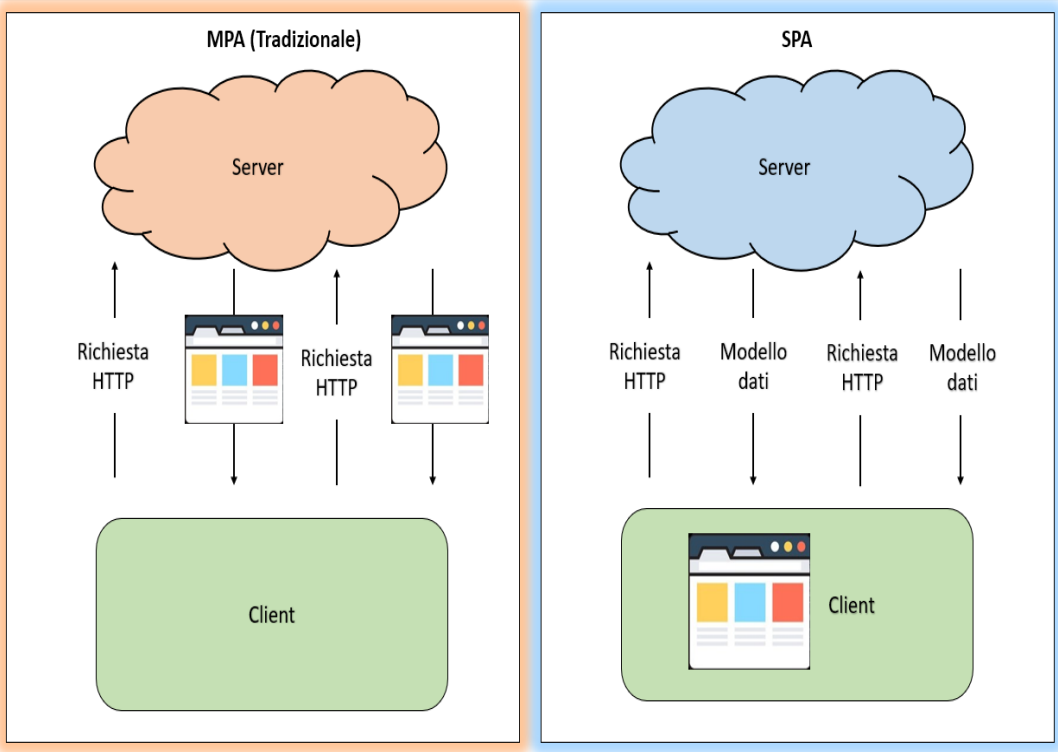

Figura 4.6: MPA vs SPA

2. Applicazioni a pagina singola (SPA) [18]: è un'applicazione il cui ciclo di vita e utilizzo è gestito all'interno di un browser e non richiede il ricaricamento della pagina durante il suo utilizzo. Alcuni esempi pratici di come si presenta questo applicativo sono: Gmail, Google Maps, Facebook o GitHub.
   Le SPA sono caratterizzate da un'unica pagina web il cui compito è di caricare tutti gli altri contenuti al primo caricamento. Si può sviluppare una applicazione di questo genere grazie a moderni framework basati su JavaScript come AngularJS, Angular, React e Vue.js.
   Gli applicativi a pagina singola aiutano a mantenere l'utente in uno spazio web confortevole dove i contenuti vengono presentati all'utente in modo semplice, facile e praticabile.

   - Vantaggi: è veloce ed efficiente, dato che la maggior parte delle risorse (HTML + CSS + script) vengono caricate solo una volta per tutta la durata del ciclo di utilizzo dell'applicazione. Il modello dei dati rappresenta

il contatto tra l'interfaccia client e il servizio REST sviluppato e viene trasmesso attraverso le azioni del protocollo HTTP.
Lo sviluppo è semplificato e il server non si occupa del rendering delle interfacce. Le SPA sono facili da testare attraverso gli strumenti di ispezione del contenuto del browser: è possibile monitorare le operazioni di rete, investigare gli elementi della pagina e i dati associati. Una SPA è in grado di funzionare anche offline.
Gli sviluppatori possono dividersi in due team: uno sviluppa il servizio REST e uno l'applicativo client. Tramite una buona documentazione, una continua comunicazione tra i team ed eventualmente implementazioni emulative temporanee (mock) è possibile parallelizzare la fase di implementazione e realizzazione del progetto in tempi ridotti.

- Svantaggi: non è un compito facile realizzare l'ottimizzazione SEO. Se un utente disabilita JavaScript nel proprio browser, non sarà possibile presentare l'applicazione e le sue azioni in modo corretto. Rispetto all'applicazione "tradizionale", la SPA è meno sicura.

Sulla base dell'analisi effettuata è stato scelto il modello SPA. I framework a disposizione per lo sviluppo di questa tipologia di applicativo sono molteplici, di seguito sono riportati alcuni dei principali presi in considerazione:

- AngularJS: [19] è un popolare framework frontend basato su JavaScript per la creazione di applicazioni web che è stato rilasciato da Google nell'anno 2010. In passato è stato molto popolare grazie alle sue ricche funzionalità integrate per creare applicazioni web di tipo reattivo.
  - Vantaggi: inizialmente era molto diffuso ed esisteva un buon supporto da parte della comunità di sviluppatori, e rappresentava una delle poche soluzioni per lo sviluppo di un applicativo web. Si basava sulla libreria jQuery per rivoluzionare la programmazione classica in JS introducendo numerose funzionalità.
  - Svantaggi: attualmente l'utilizzo di JavaScript implica dover possedere una profonda conoscenza del linguaggio e delle sue potenzialità, la sintassi è decisamente complessa e si presta facilmente alla creazione di numerosi errori. Inoltre, è poco intuitivo e risulta avere un livello debole di astrazione. Quindi è diventato, con l'avvento dei nuovi moderni framework, una soluzione poco efficiente e produttiva.
    Un esempio è il limite del two-way data binding: vista e modello sono sincronizzati in maniera automatica, ma questo meccanismo riduce le prestazioni quando il modello cresce e, lavorando in ambito SPA, i memory leak sono frequenti. Il supporto della comunità e di nuovi aggiornamenti sono stati progressivamente abbandonati.
    Il suo successore in casa Google è Angular versione 2+.
- Angular (2+): [20] è il nuovo framework sviluppato da Google nel 2016, rappresenta una soluzione versatile in quanto rende possibile la creazione di applicazioni web, native o desktop. Si basa sull'utilizzo di TypeScript (TS), un

linguaggio semplice e leggero, che permette al programmatore di evitare la complessità del mondo JavaScript.
Rispetto al suo predecessore non ha praticamente nulla in comune e rappresenta una vera e propria rivoluzione e modernizzazione. Alcune delle pietre miliari di questa tecnologia sono DC (Dependency injection), Ajax, l'utilizzo di HTML e CSS.

  - Vantaggi: TS è un linguaggio type safe, sono previsti strumenti in grado di supportare il linguaggio in fase di refactoring, nell'autocompletamento o nella navigazione del codice. Infine, per quanto riguarda TypeScript, è incluso un supporto alla creazione e gestione di interfacce e classi come nella programmazione tradizionale orientata agli oggetti.
    Il framework risulta quindi molto versatile, garantisce elevate velocità e performance. Le applicazioni sviluppate si basano sul principio di modularità, questo permette il riutilizzo del codice. Il codice generato è decisamente inferiore ad altri framework e questa caratteristica dona chiarezza e leggibilità al software.
    Attraverso lo strumento Karma, dedito ai test, è possibile risalire a comuni errori implementativi garantendo una buona stabilità al progetto.
    Bisogna sottolineare come la comunità di sviluppatori stia crescendo con la progressiva migrazione a questa moderna tecnologia. Questo permette l'utilizzo di componenti creati da altri programmatori con un determinato obiettivo e permette di ridurre i tempi per la creazione dell'applicativo.
  - Svantaggi: esistono soluzioni più flessibili, il framework ha alcune regole e strutture fisse da seguire durante la fase di sviluppo, le quali richiedono una preparazione e una confidenza relativa all'ambiente da parte del programmatore. La curva di apprendimento è considerata da alcuni sviluppatori la più ripida delle soluzioni mostrate in quanto bisogna imparare alcune caratteristiche tipiche del framework per permetterne il corretto funzionamento.

- React: [21] è una tecnologia di Facebook che si basa sul moderno linguaggio di programmazione JSX, il quale esegue un processo di ottimizzazione durante la fase di compilazione del codice sorgente JavaScript. Il codice generato viene quindi eseguito più velocemente di un codice equivalente scritto direttamente in JS.
  La sua peculiarità è quella di permettere l'individuazione di errori in fase di progettazione piuttosto che in compilazione. JSX svincola gli sviluppatori dal sistema di ereditarietà basato sul prototipo fornito da JavaScript. Le espressioni e le istruzioni, tuttavia, sono per lo più uguali a JS, quindi è facile per i programmatori iniziare a utilizzarlo. Infine, va sottolineato come questo framework a differenza di quelli analizzati fino ad ora si basa sul concetto di DOM virtuale:

    > Il DOM virtuale è un'astrazione del DOM HTML. È leggero e scollegato dai dettagli di implementazione specifici del browser. Poiché il DOM stesso era già un'astrazione, il DOM virtuale è, in realtà, un'astrazione di un'astrazione. [22]

  - Vantaggi: è un framework caratterizzato dalle alte performance e flessibile. Consente una distribuzione più agile del codice. Esiste un ottimo supporto della comunità di sviluppo la quale mette a disposizione una serie di pacchetti integrabili nel proprio progetto ai fini di renderne più veloce l'implementazione.
    Rende possibile l'ottimizzazione SEO, si basa sul meccanismo del riutilizzo dei componenti e risulta facile implementare delle unità di test.
    Il data binding unidirezionale garantisce una solida struttura evitando errori comuni al programmatore, mentre l'utilizzo del DOM virtuale permette una gestione più facile ed efficiente del contenuto dell'interfaccia.
    Infine, è caratterizzato da una buona scalabilità e versatilità (utilizzato sia per lo sviluppo di applicazioni web sia di applicazioni native).
  - Svantaggi: il team di Facebook ha raccomandato di utilizzarlo insieme all'architettura Flux, quindi è soggetto a questo vincolo progettuale per ottenere una soluzione dalle alte prestazioni. In generale non è tra i framework più facili da imparare in quanto molto verboso. Inoltre, l'ambiente è in continuo aggiornamento e richiede un costante adeguamento da parte dei programmatori. La documentazione per via di questo problema tende ad essere minimale e difficile da realizzare.

- Vue.js: [23] è un framework progressivo destinato allo sviluppo di interfacce utente. La libreria principale si concentra unicamente sul livello di visualizzazione ed è facile da integrare e arricchire con altre librerie o progetti esistenti. Nonostante ciò, è anche perfettamente in grado di alimentare sofisticate applicazioni a pagina singola se combinato con apposite librerie di supporto e moderni strumenti di sviluppo. Anche Vue.js si basa sull'utilizzo del DOM virtuale.

  - Vantaggi: il vantaggio principale è sicuramente la sua semplicità, inoltre è caratterizzato da una progressività e buona scalabilità. La curva di apprendimento è meno ripida rispetto alle altre tecnologie mostrate e il tutto risulta decisamente flessibile.
  - Svantaggi: questo framework è posseduto da una singola persona, il suo ideatore e sviluppatore principale Evan You. Il team di mantenimento e supporto è quindi molto ridotto. L'elevata flessibilità può portare a numerosi rischi e problematiche durante la creazione del progetto.

Sulla base delle considerazioni effettuate è possibile affermare che AngularJS rappresenta un vicolo cieco dal punto di vista dello sviluppo, privo di un supporto futuro e caratterizzato da una certa obsolescenza e prestazioni minori rispetto a tutte le altre soluzioni.
Vue.js, nonostante sia un'ottima proposta, risulta ancora una scelta incerta per via di un team molto piccolo dedito al suo mantenimento e attualmente la sua diffusione è ancora limitata.
Tra quelle che possono essere definite le due tecnologie dominanti è stato scelto

come framework di riferimento Angular rispetto a React in quanto basato sulla programmazione ad oggetti, caratterizzato da un'ottima documentazione sempre aggiornata e da una particolare attenzione per alcuni aspetti di sicurezza che aiutano i programmatori a sviluppare un codice privo di errori comuni.

Capitolo 5

Architettura

La piattaforma sviluppata si basa sull'utilizzo della tecnologia di virtualizzazione Docker che permette di ospitare diverse applicazioni in grado di comunicare tra di loro. Si possono individuare, come mostrato in Figura 5.1, tre principali elementi costitutivi del framework:

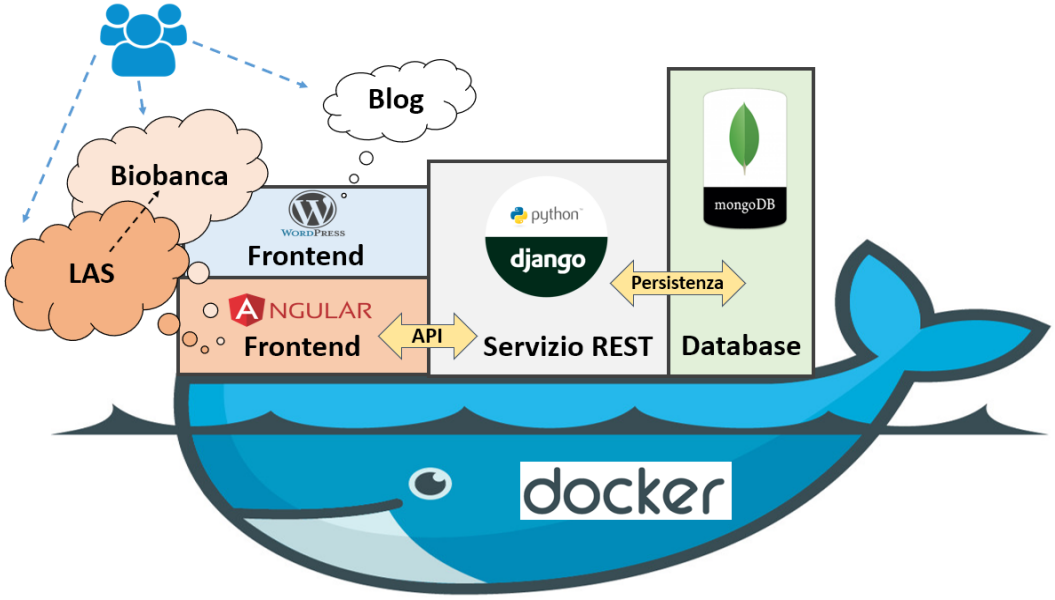

Figura 5.1: Architettura generale del nuovo framework

- Database: utilizza MongoDB per implementare la persistenza del modello dati del sistema;
- Servizio REST o Backend: rappresenta l'intero servizio sviluppato basandosi sul framework Django, espone diverse API al frontend e comunica direttamente con il Database per effettuare operazioni in lettura e scrittura dei dati;
- Frontend: è l'applicativo utilizzato direttamente dagli utenti finali. E' costituito da tre principali elementi:

- Blog: un sito sviluppato con il CSM (Content Management System) Wordpress, che fornisce alcune informazioni base del sistema a scopo divulgativo;
- Biobanca: permette l'accesso a bio-dati raccolti, analisi e statistiche (sviluppato utilizzando Angular Framework);
- LAS: è l'applicativo utilizzato dai ricercatori in laboratorio per conseguire obiettivi di uno o più progetti (come per la Biobanca sviluppato utilizzando Angular Framework).

5.1 Docker

La creazione di servizi distribuiti richiede la progettazione e messa in campo di un certo numero di componenti software eterogenei.
Docker è una tecnologia open source utilizzata per la costruzione, distribuzione e rilascio di applicazioni basate sul meccanismo della virtualizzazione a livello di contenitore.
I suoi quattro componenti principali sono:

- Docker Engine: come mostrato in Figura 5.2 rappresenta il cuore della piattaforma, nonché il processo demone eseguito in background sulla macchina fisica che deve ospitare il container. Fornisce l'accesso a tutte le funzionalità e i servizi messi a disposizione da Docker.
- Dokcer Client: è l'interfaccia per le API esposte da Dokcer Engine. Mette a disposizione una CLI (Command line interface) con un set di comandi che iniziano tutti con la parola chiave "docker".
- Docker Image: è un insieme di file e parametri che definisce e configura un'applicazione da usare a runtime. Le immagini sono organizzate in livelli stateless e immutabili. Ogni strato del file system dell'immagine è accessibile in sola lettura. In caso di conflitti relativi al contenuto dei diversi livelli di una immagine, viene risolto il conflitto considerando lo strato più alto come il vincitore.
- Docker Container: è un'istanza in esecuzione di un'immagine il cui file system viene arricchito da un ulteriore ultimo strato a cui è possibile accedere sia in lettura sia in scrittura. Questo strato viene distrutto con la distruzione del contenitore, è necessario quindi effettuare un'operazione di commit per salvarne le modifiche rendendolo uno strato di tipo read-only dell'immagine di origine. I contenitori possono essere associati a dei volumi, i quali rappresentano un modo semplice e immediato per condividere parte del file system con la macchina host. Le operazioni eseguite sul volume sono interamente controllate dal driver della macchina ospitante e il suo contenuto è mantenuto anche in caso di rimozione del contenitore. Infine, i container possono essere organizzati in reti e comunicare tra loro. La comunicazione può avvenire sia tra contenitori che tra il mondo esterno e un contenitore, per questo motivo viene effettuato il mapping tra una porta della macchina host e una porta del contenitore, abilitando in questo modo le connessioni in ingresso.

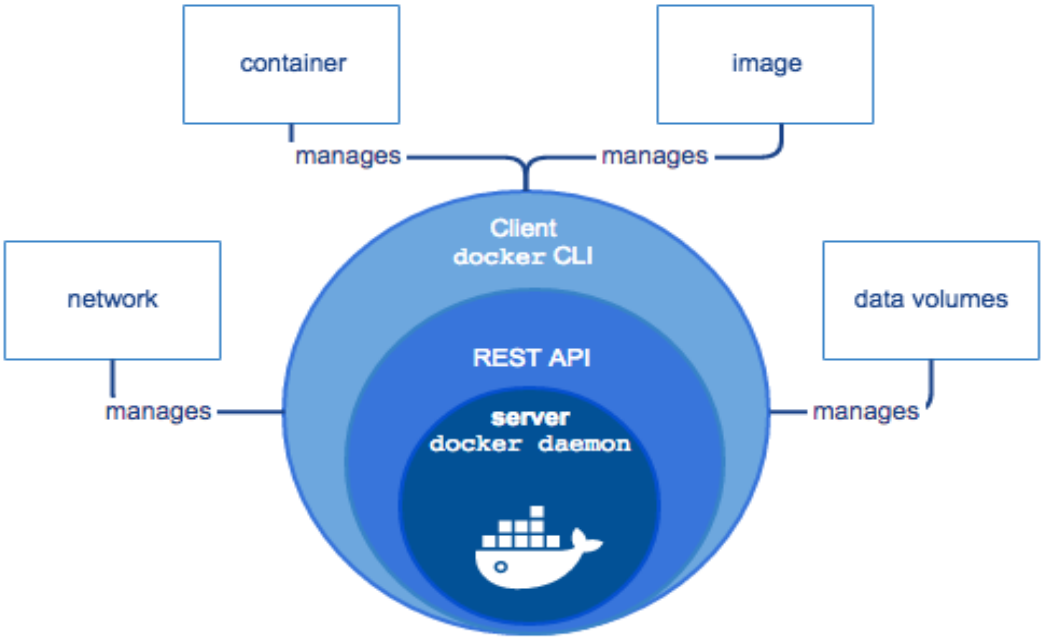

Figura 5.2: Docker

Per permettere la comunicazione tra contenitori sulla stessa macchina vengono create apposite configurazioni di rete:

1. None: il contenitore possiede solamente l'interfaccia di loopback;
2. Bridge: i contenitori appartenenti a questa configurazione di rete possono comunicare mediante l'utilizzo di indirizzi IP o il loro nome;
3. Host: permette al contenitore di accedere all'intero stack di rete della macchina ospitante.

Inoltre, è possibile instaurare una comunicazione tra contenitori di macchine host separate tramite la configurazione di Docker Swarm.

Docker consente una vera indipendenza tra le applicazioni e le infrastrutture e, tra gli sviluppatori e gli operatori IT, al fine di sbloccare il loro potenziale e creare un modello per una migliore collaborazione e innovazione.
In particolare, per lo sviluppo e la futura distribuzione del progetto è stato utilizzato Compose, il quale è uno strumento per la definizione e l'esecuzione di applicazioni Docker multi-contenitore. Questo approccio ha permesso di creare un sistema così composto:

- Biocanca dati
- Blog

- Laboratory Assistant Suite
- Servizio REST
- Databases

Una delle destinazioni finali del progetto è quello di poter condividere e distribuire questo applicativo in modo tale che ogni organizzazione e associazione possano impostare, personalizzare e utilizzare la piattaforma sviluppata.

5.1.1 Portainer

L'utilizzo di Docker richiede spesso la gestione delle immagini e dei container tramite linea di comando durante le fasi di sviluppo, test, installazione e collaudo. Per migliorare l'esperienza sia degli sviluppatori sia degli amministratori di sistema è stato previsto l'utilizzo di Portainer.

[24] Portainer è uno strumento open-source che fornisce una interfaccia utente la quale consente di gestire facilmente l'host Docker o eventualmente il cluster Swarm, evitando la lettura dei log classici i quali risultano spesso complicati e meno user-friendly. Questa tecnologia è pensata per essere semplice. Consiste in un singolo contenitore che può essere eseguito su qualsiasi motore Docker (sono supportati Docker per Linux e Docker per Windows). Risulta quindi possibile gestire stack, contenitori, immagini, volumi, reti nonché tutto l'ambiente creato con la tecnologia Docker.

Nel caso specifico di questo progetto, basandosi su una soluzione di tipo cluster attraverso l'impiego di Docker Swarm, è ideale installare e utilizzare Portainer sui nodi Manager del cluster.

5.2 Frontend

Per quanto riguarda lo sviluppo della parte client sono stati creati un Blog, il cui scopo è fornire un supporto informativo e divulgativo, e due differenti applicazioni web, rispettivamente una per la gestione del LAS e una per la gestione della Biobanca.

Un'applicazione web permette di utilizzare tecnologie e linguaggi differenti costruendo un frontend semplice e universale su tutte le piattaforme in grado di essere mantenuto nel tempo.
La scelta è ricaduta sul modello "Single Page Web Application" (SPA) il quale permette di recuperare tutto il codice utile durante la fase di caricamento della singola pagina ed eventualmente di reperire le altre risorse in maniera dinamica quando necessario. La SPA, come mostrato in Figura 5.3, rivoluziona lo schema MVC delle applicazioni web tradizionali. Il vantaggio principale è quello di farsi carico di una parte della logica di business gestendo le interazioni tra utente e Servizio REST con un proprio schema architetturale Modello-Vista-Controllore. Il DOM è generato quindi sulla macchina dell'utente finale mentre il punto di comunicazione tra

frontend e backend, il quale espone le API, è rappresentato dal Modello Dati. Tale modello viene incapsulato e scambiato attraverso il protocollo applicativo HTTP.

Durante la fase di sviluppo si è cercato di creare le basi per una possibile evoluzione futura ad Applicazione Web Progressiva introducendo il principio "offline first": l'applicativo diventa in grado di operare anche in assenza di connessione di rete, nel nostro caso effettuando una prima memorizzazione delle modifiche apportate al Modello Dati a livello locale nella cache della macchina dell'utente e successivamente una memorizzazione sincrona con il server qualora la disponibilità di rete sia effettiva.

Per lo sviluppo e il mantenimento del blog è stato utilizzato Wordpress basato su un database MySQL, mentre per l'implementazione dell'applicativo relativo al LAS e alla Biobanca è stato utilizzato Angular.

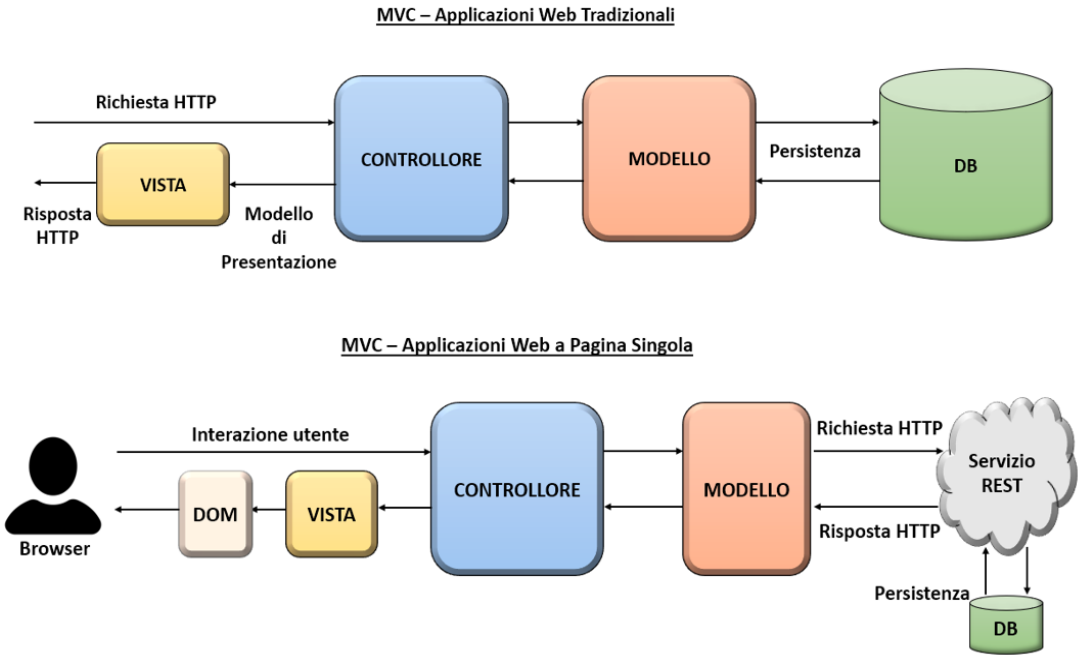

Figura 5.3: Evoluzione del Model-View-Controller

5.2.1 Angular

Angular è una piattaforma utilizzata principalmente per la creazione di applicazioni web, mobile e desktop sviluppata e mantenuta da Google LLC. Per convenzione, con Angular si intende la versione del framework dalla numero 2 in poi rilasciata dal 15 settembre 2016, mentre con AngularJS si intende la versione numero 1. Tale versione si basava su JQuery e aveva la caratteristica di essere più lenta e poco reattiva per determinate funzionalità. La dicitura Angular 2+ invece è utilizzata per indicare le ultime versioni del nuovo framework. La versione di Angular utilizzata durante lo sviluppo di questo progetto è l'ultima stabile ovvero la distribuzione 5.2.10.

Angular si basa sostanzialmente su TypeScript (TS), il quale è un linguaggio super-set di JavaScript, creato da Microsoft e che basa le sue caratteristiche su ECMA-Script 6 (ES6). Una delle sue peculiarità è quella di essere "type safe" (il tipo di

dato è opzionale), si basa su interfacce, classi, moduli e prevede la sintassi per le "arrow function".
Il problema principale di TypeScript è che ancora non tutti i browser supportano ES6, quindi è necessario compilare tutto il codice sviluppato, per trasformarlo (operazione chiamata "transpiling") in un linguaggio interpretabile dagli attuali browser con lo stesso livello di astrazione, ossia ECMAScript 5 (ES5).

Angular permette, basandosi su TS, HTML e CSS, di creare un'applicazione suddivisa in moduli dove ognuno a sua volta è formato da diversi componenti e servizi in grado di permettere lo scambio dei dati, sia in locale sia in remoto verso il server.
L'intera piattaforma si basa quindi sul principio di modularità e sul meccanismo del riutilizzo. Un vantaggio infatti è sicuramente quello di riuscire a progettare determinati componenti che potranno essere riadattati in base alle necessità in diversi moduli applicativi, permettendo quindi il riutilizzo di codice utile in diverse parti del progetto diminuendo il tempo necessario per lo sviluppo.

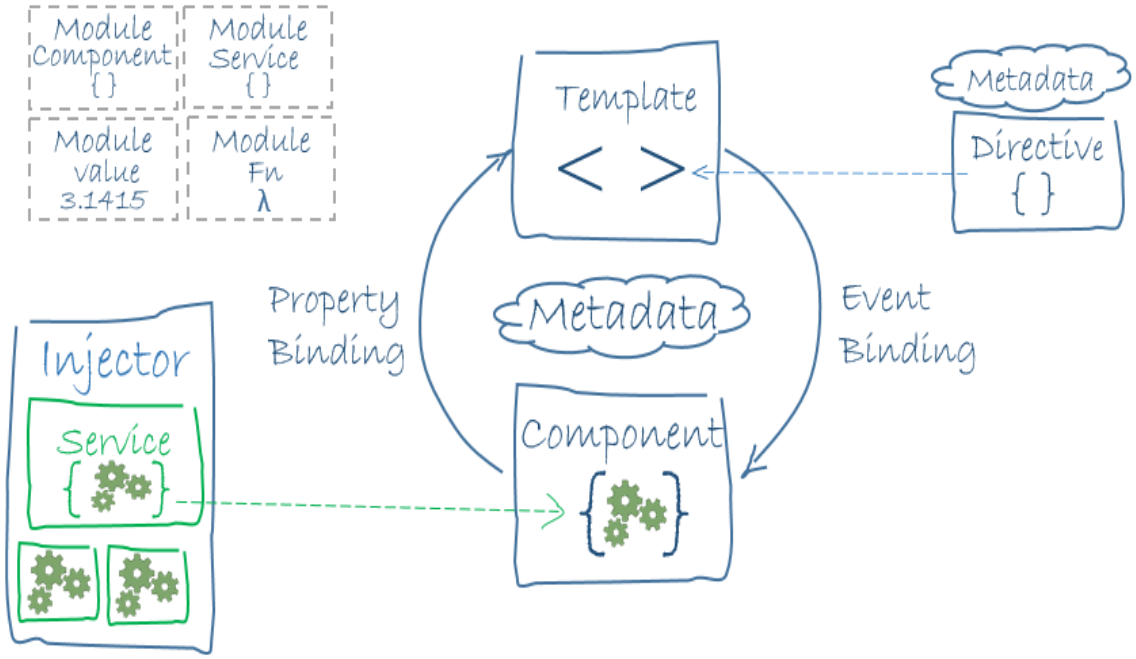

Figura 5.4: Architettura Angular Framework

L'architettura generale di questo framework, come mostrato in Figura 5.4, si basa sui seguenti elementi:

- Modulo: rappresenta l'unità fondamentale. All'interno di esso vivono componenti e servizi. Un'applicazione è sviluppabile in maniera modulare: il modulo principale viene caricato durante il primo avvio mentre tutti gli altri moduli vengono caricati su richiesta attraverso il meccanismo del lazy-loading. Suddividendo il progetto in questa maniera risulta più facile la sua gestione e rende più fluida la sua esecuzione a runtime.

- Componente: vive all'interno di un modulo ed è caratterizzato da un controllore contenente la business logic implementata attraverso un file scritto in TS e

da un template scritto utilizzando HTML, CSS e direttive. La comunicazione tra questi due elementi avviene principalmente grazie al property binding e all'event binding. Esistono principalmente due tipologie di componente:

  - Smart: è un componente padre il cui compito è quello di reperire i dati utilizzando appositi servizi e di fornirli ai propri componenti figlio. I dati sono passati tramite il meccanismo di property binding.
  - Dummy: è il componente figlio e il suo unico scopo è quello di effettuare il rendering dei dati ricevuti. Può comunicare coi componenti padre tramite il meccanismo di event binding.

- Servizio: è un oggetto costituito da diversi metodi che offrono diverse funzionalità. Ogni servizio viene per prima cosa creato, poi dichiarato in un modulo e infine iniettato nei suoi componenti di tipo Smart che necessitano di accedere alle sue funzionalità.

- Direttiva: sostanzialmente sono un semplice attributo HTML a cui possono essere assegnate espressioni che rappresentano il comportamento desiderato. Le direttive di tipo attributo hanno il compito di modificare l'aspetto o il comportamento di un elemento mentre le direttive di tipo strutturale sono in grado di modificare la struttura del DOM aggiungendo o rimuovendo elementi.

5.2.2 Redux

Poiché i requisiti per le applicazioni a pagina singola sono diventati sempre più complessi e articolati, è necessario mantenere informazioni di stato lato client. Questo stato può includere risposte del server e dati memorizzati nella cache, nonché dati creati localmente che non sono ancora stati inoltrati al server.
Durante la fase di sviluppo di un applicativo frontend il programmatore si fa carico di dover progettare una macchina a stati dove le sequenze e le combinazioni di azioni e/o eventi generati dall'utente sono molteplici e spesso imprevedibili.
Redux, il cui modello è mostrato in Figura 5.5, è un contenitore JavaScript il quale tenta di rendere prevedibili le mutazioni di stato imponendo alcune restrizioni su come e quando possono verificarsi gli aggiornamenti.

Queste restrizioni si riflettono nei tre principi di Redux [25]:

- Unica fonte di verità: lo stato dell'intera applicazione è memorizzato in una struttura ad oggetti all'interno di un singolo archivio (Store). Tale struttura è un albero di diversi sotto-stati che vengono definiti dal programmatore e che evolvono nel tempo (in Figura 5.6 è mostrato l'esempio dell'evoluzione dello stato dell'applicazione sulla base del processo di autenticazione);

- Sola lettura: per determinare un cambiamento dello stato è necessario emettere un'azione, ovvero un oggetto che descriva cosa sia successo. Si limita la funzionalità di viste e servizi, i quali non possono modificare direttamente l'albero degli stati ma solamente leggerlo o esprimere l'intento di trasformarlo

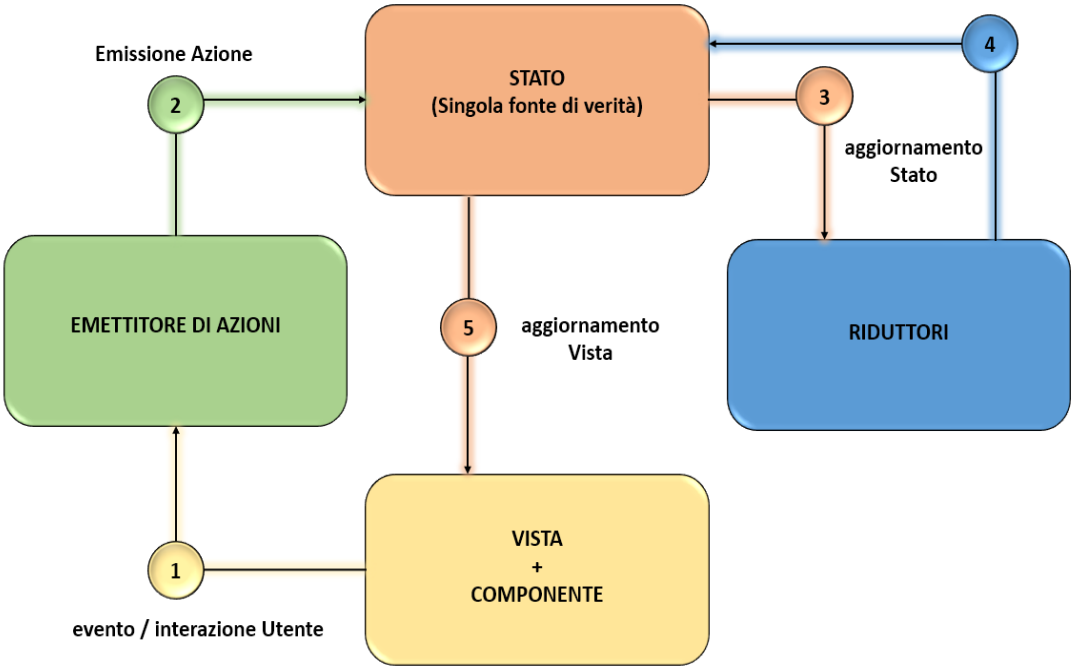

Figura 5.5: Modello Redux

tramite l'emissione di azioni. Queste azioni sono semplici oggetti, possono essere registrate, serializzate, memorizzate e successivamente riprodotte a scopo di debug o di test;

- Funzioni pure: per elaborare un'azione emessa si utilizzano i riduttori, i quali sono funzioni pure che elaborano lo stato precedente e l'azione emessa restituendo infine lo stato successivo. E' possibile suddividere l'applicazione in gruppi diversi di riduttori che gestiscono parti specifiche dell'albero di stato.

5.2.3 Meccanismo di Cache

Per questo genere di applicazioni un ruolo fondamentale è assunto dalla Cache.
Sostanzialmente, è possibile che il client memorizzi localmente informazioni di stato come affermato in precedenza, in modo tale da poterle reperire quando necessario e permettere così l'evoluzione del sistema. Inoltre, è da sottolineare l'importanza del salvataggio in locale di dati derivanti da operazioni che sono coinvolte nella logica di business principale dell'applicativo.
La Cache permette quindi di lavorare in locale riuscendo a popolare eventualmente viste con dati salvati in precedenza e successivamente è in grado anche di memorizzare le modifiche effettuate su tali dati per poi effettuare una sincronizzazione futura con il server.
Successivamente sono riportate le tecnologie disponibili per il salvataggio di dati nel browser, ognuna con una breve descrizione.

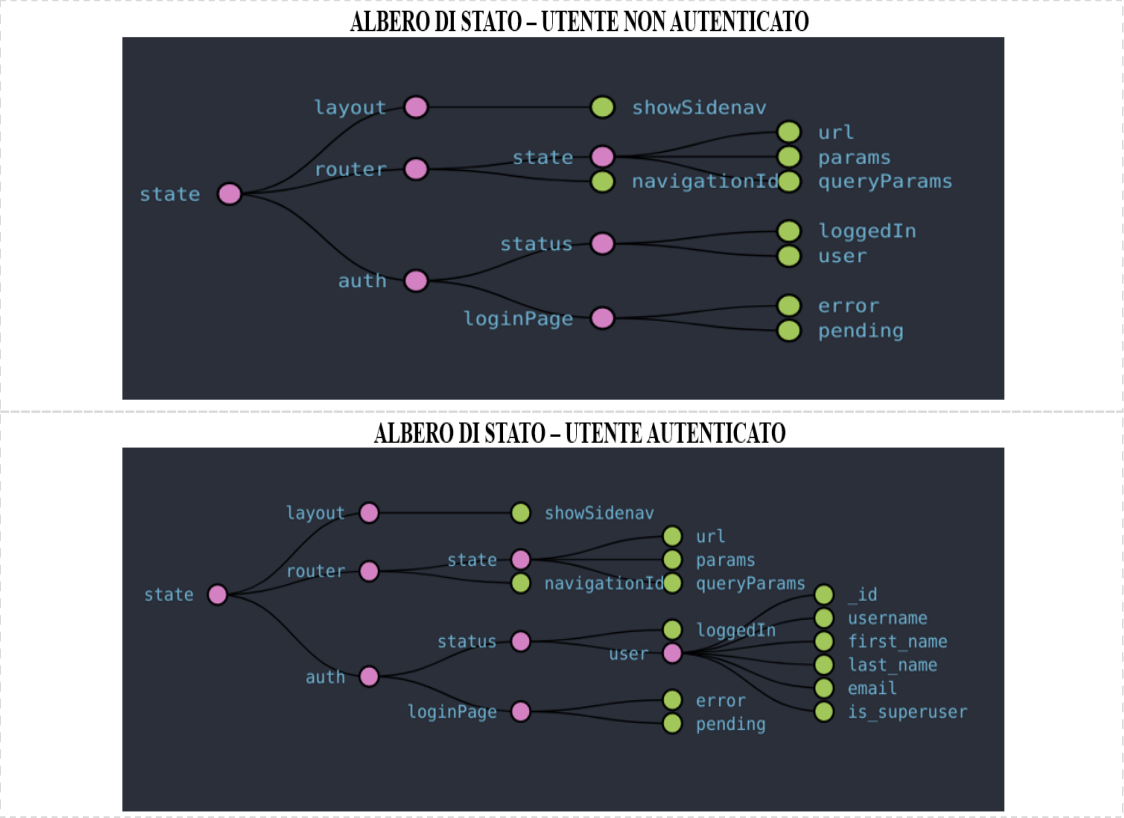

Figura 5.6: Evoluzione dell'Albero di Stato

Cache Storage

Alcuni browser moderni supportano la nuova API Cache. Questa è progettata per archiviare le risposte HTTP corrispondenti a richieste specifiche. Queste potrebbero essere richieste e risposte regolari create durante l'esecuzione dell'applicazione, oppure potrebbero essere create esclusivamente allo scopo di memorizzare alcuni dati nella Cache.
Il suo utilizzo risulta quindi molto utile per la memorizzazione di risorse DOM di siti Web in assenza di connessione, in modo che il sito possa essere successivamente utilizzato senza una connessione di rete.
Questa tipologia di storage nel progetto non è stata utilizzata direttamente, tuttavia se ne è valutato un possibile utilizzo futuro. Il suo utilizzo è tipico nelle applicazioni web progressive anche se attualmente non tutti i moderni browser lo supportano.

Cookies

I cookies memorizzano spesso le impostazioni di preferenza, come per esempio la lingua o la posizione. Quando si ritorna in navigazione su una determinata pagina, permettono di personalizzarla sulla base dei parametri memorizzati rispondendo a diverse esigenze degli utenti.
Un cookie può essere modificabile dal codice sorgente lato client, mentre se è di tipologia "HTTP Only" potrà essere solo impostato dal servizio REST e non sarà

quindi possibile accederci dall'applicazione.
Altri attributi di interesse di un cookie sono sicuramente "Secure" che indica che lo scambio di questo avverrà solo attraverso connessioni HTTP criptate, mentre "SameSite" consente di inviare il cookie all'applicazione solo se la richiesta proviene dallo stesso dominio.
Uno dei limiti è rappresentato dalla quantità di dati che possono memorizzare, ovvero: 4 KB. Bisogna considerare il fatto che possono essere disabilitati attraverso le impostazioni del browser.

Indexed DB

L' Indexed DB, rispetto a tutte le altre tipologie mostrate in questa breve analisi, è stato progettato per funzionare con quantità di dati maggiori.
Fornisce sia un'API di tipo sincrono sia una di tipo asincrono. In ambito pratico, tuttavia, quasi tutte le implementazioni sono asincrone.
Inoltre, Indexed DB, come rivela il nome, fornisce il supporto all'utilizzo di indici. Si possono quindi eseguire interrogazioni base sul database associato al dominio dell'applicazione web e recuperare i record sulla base del valore delle chiavi. Supporta anche le transazioni e permette l'utilizzo di tipi di dato semplici (ad es. il tipo Date).
Questa tipologia di storage è stato utilizzato durante lo sviluppo del progetto per permettere la memorizzazione di meta-informazioni relative all'utente e alle sue attività svolte.
Il suo utilizzo principale è sicuramente quello relativo al salvataggio di grandi quantità di dati che possano permettere il funzionamento e l'operatività dell'applicativo anche in assenza di connessione.
Un altro aspetto da sottolineare è la garanzia di alte prestazioni durante l'utilizzo di questa tecnologia, dovute certamente al modello asincrono e alla possibilità di utilizzare gli indici in fase ricerca.

Local Storage

Il Local Storage invece è stato progettato per piccole quantità di dati.
Si basa su di un'API sincrona la quale permette essenzialmente una memorizzazione di coppie chiave e valore.
Rispetto all'Indexed DB risulta molto più semplice e immediato e quindi viene utilizzato per memorizzare meta-dati dell'applicazione, il rischio che ne consegue da uno scorretto utilizzo è di rallentare o addirittura bloccare l'applicativo dato che si basa su un meccanismo sincrono.

Session Storage

Consente di avere un meccanismo di persistenza dei dati distinto per ogni sessione di navigazione in ogni finestra di lavoro del browser. L'apertura di una nuova finestra implica l'apertura di una nuova sessione, il che differisce da come funzionano i cookie di sessione.
Rispetto al Local Storage i dati vengono cancellati con la chiusura della sessione.

La sessione della pagina dura fino a quando il browser è aperto oppure se sopravvive alla ricarica della pagina o al suo ripristino.
Questa ultima tipologia non è stata utilizzata durante lo sviluppo del progetto, non se ne escludono usi futuri, anche se attualmente è in contrasto con il meccanismo base degli applicativi a pagina singola sviluppati.

5.2.4 WordPress

WordPress [26] è un CMS (Content Management System) open source, ovvero uno strumento software, installato su un server remoto, il quale offre un'interfaccia attraverso la quale l'amministratore di sistema può gestire l'intero sito web.
Il database di riferimento è MySQL. Permette di salvare al suo interno tutte le informazioni relative al sito sviluppato come immagini, testo e molto altro ancora.

Uno dei punti forza di questa tecnologia è la caratteristica di avere un'ottima comunità di supporto che ne permette la condivisione di contenuto e la risoluzione di problemi.
WordPress si basa sul meccanismo dei plugin (gratuiti o a pagamento): moduli che estendono le funzionalità dell'applicativo e aggiungono nuove caratteristiche ed elementi ai siti realizzati.

Il compito di WordPress, nello specifico del progetto, è di fornire quindi un semplice supporto per lo sviluppo e la gestione del Blog. Durante il processo di distribuzione del framework, gli usufruenti, potranno investire risorse limitate ma efficaci per la gestione e personalizzazione del proprio sito web. La versione utilizzata per la realizzazione del progetto è Wordpress:Php7.2-fpm-alpine disponibile sull'Hub di Docker.

5.3 Backend

Per quanto riguarda il backend si è sviluppato un servizio di tipo Representational State Transfer (REST). Il servizio garantisce la possibilità di poter accedere al contenuto di risorse web, ogni risorsa è unica e indirizzabile usando la sintassi universale Uniform Resource Locator (URL).
REST è una tipologia di architettura software per sistemi ipermediali distribuiti di tipo client/server che si basa sul protocollo applicativo HTTP. I client inoltrano richieste ai server che a loro volta processano le richieste e restituiscono una risposta. Richieste e risposte contengono una rappresentazione delle risorse.
In passato era il server stesso che, sulla base delle richieste del client, generava il contenuto di una pagina web; con l'avvento delle applicazioni a pagina singola la generazione del DOM e parte della logica di business sono stati spostati nel frontend. In questo modo il server espone il proprio servizio tramite delle API pubbliche al client. In questo nuovo modello il client e il server si scambiano una rappresentazione delle risorse mappate dal servizio REST.
Nel nostro caso specifico la rappresentazione delle risorse è un contenuto di tipo JSON conforme con la rappresentazione dei dati presenti nel database o inseriti dall'utente durante il normale utilizzo dell'applicativo.

Uno dei principi di questa architettura è l'assenza di stato: la comunicazione client–server è vincolata in modo che nessun contesto client venga memorizzato sul server tra le richieste.
Il servizio REST sviluppato permette di mappare risorse sia per il LAS sia per la Biobanca.

5.3.1 Django Framework

Django è un framework open source basato sul linguaggio di programmazione Python. Rappresenta un ottima scelta in quanto tra le caratteristiche principali troviamo la velocità, la scalabilità, un'ottima flessibilità e una diversificata collezione di librerie che permettono di evitare errori di sicurezza comuni.
La scalabilità è di fondamentale importanza per creare un sistema distribuito: all'interno del framework troviamo per esempio meccanismi di caching. Un altro principio fondante è la modularità che lo rende adattabile alle esigenze dello sviluppatore. La programmazione modulare permette di poter usufruire di codice sviluppato da terze parti, permettendone l'inserimento nel progetto in maniera semplice e trasparente. In caso di manutenzione o aggiornamento del codice è possibile quindi agire solo su un determinato modulo senza sconvolgere l'assetto dell'intero sistema.
È pensato per rendere la progettazione di applicazioni più agile, sia che si tratti di applicazioni modeste o molto complesse, grazie ai numerosi strumenti già integrati che guidano il programmatore durante la fase di sviluppo.
La comunità di utilizzatori del progetto è molto diffusa e diversificata sulla rete, si ha quindi accesso a diverse librerie e plugin sviluppati da esperti del settore che permettono di risolvere problemi comuni e semplificare lo sviluppo del software.
In particolare per questo progetto si è utilizzato Django 2.0.7 con Python 3.6.

5.3.2 MongoDB

MongoDB è un database NoSQL le cui principali caratteristiche sono di essere orientato al documento, di avere elevate prestazioni, di garantire alta disponibilità e di presentare una scalabilità facilitata.
Sulla base del Teorema CAP citato in precedenza, garantisce la consistenza e la tolleranza di partizione.
I dati vengono salvati in documenti JSON (resi persistenti in forma binaria: BSON).
Questo database NoSQL fornisce un modello dati versatile che può essere integrato con i tipi di dato nativi dei linguaggi di programmazione più utilizzati. Le relazioni tipiche del modello relazionale classico sono modellabili in questo database NoSQL mediante due tecniche differenti:

- Documenti innestati: ogni documento è in grado di contenere sotto-documenti (in questo modo le interrogazioni sono ottimizzate). Ogni documento radice possiede il contenuto dei propri sotto-documenti, che sono quindi facilmente reperibili e analizzabili. Questo approccio è utilizzato principalmente per esprimere relazioni di tipo uno a uno o relazioni di tipo uno a molti. Un limite di questo approccio è rappresentato dalla dimensione massima di 16 MB per documento.

- Riferimenti: in maniera molto simile alle tabelle dei database relazionali si memorizza un id univoco in ogni documento. Se è necessario far riferimento a quel determinato documento all'interno di un altro, si inserirà l'id in un campo del JSON (al posto dell'intero documento come visto nella tecnica precedente). Questa tecnica è onerosa in fase di ricerca e viene utilizzata tipicamente quando le relazioni di tipo molti a molti rischiano di far crescere a dismisura i documenti.

Le caratteristiche principali di questo database sono:

- Aggregazione: esiste il supporto per le aggregazioni di dati di tipo MapReduce e di tipo Aggregration Framework.

- Affidabilità: il carico di lavoro può essere aumentato e reso sempre affidabile tramite l'impiego di replica set. Un replica set consiste in copie multiple dei dati. Le repliche sono suddivise in repliche di tipo primario o secondario. La replica primaria effettua tutte le scritture e le letture mentre le repliche secondarie mantengono una copia dei dati della replica primaria attraverso un meccanismo di replicazione incluso nel prodotto. In caso di fallimento della replica primaria un algoritmo determina quale replica secondaria è adatta a sostituirla. Le copie secondarie hanno la possibilità di effettuare letture, con dati eventualmente consistenti di default.

- File Storage: il database può essere utilizzato come un file system, traendo vantaggio dalla caratteristiche di replicazione e di bilanciamento su più server per memorizzare file, anche di grandi dimensioni. La funzione di riferimento è GridFS, usata, ad esempio, nei plugin di NGINX. Il file viene scomposto in tante parti più piccole, chiamate chunks. Ogni chunk viene salvato su un documento separato.

- Indicizzazione: ogni campo in MongoDB può essere indicizzato per ottimizzare interrogazioni di ricerca.

- Interrogazioni ad hoc: le ricerche sono effettuabili per campi, intervalli ed espressioni regolari. Le query possono restituire uno o più campi specifici del documento oppure includere funzioni scritte dal programmatore in JS.

- Sharding: i dati delle collezioni vengono distribuiti sui diversi nodi disponibili (una chiave di shard viene scelta dall'amministratore). I dati, come mostrato nella Figura 5.7 sono suddivisi in intervalli (basati sulla chiave di shard) e distribuiti su molteplici shard. Ogni shard sostanzialmente è un replica set, formato da una replica primaria e due o più repliche secondarie. Questo meccanismo permette di scalare orizzontalmente. E' previsto, inoltre, un meccanismo di bilanciamento dei dati all'interno del cluster: gli intervalli dei dati vengono trasferiti da uno shard troppo carico ad uno shard meno carico.

La versione utilizzata 4.0 è stata rilasciata a Giungo 2018 e introduce alcune novità:

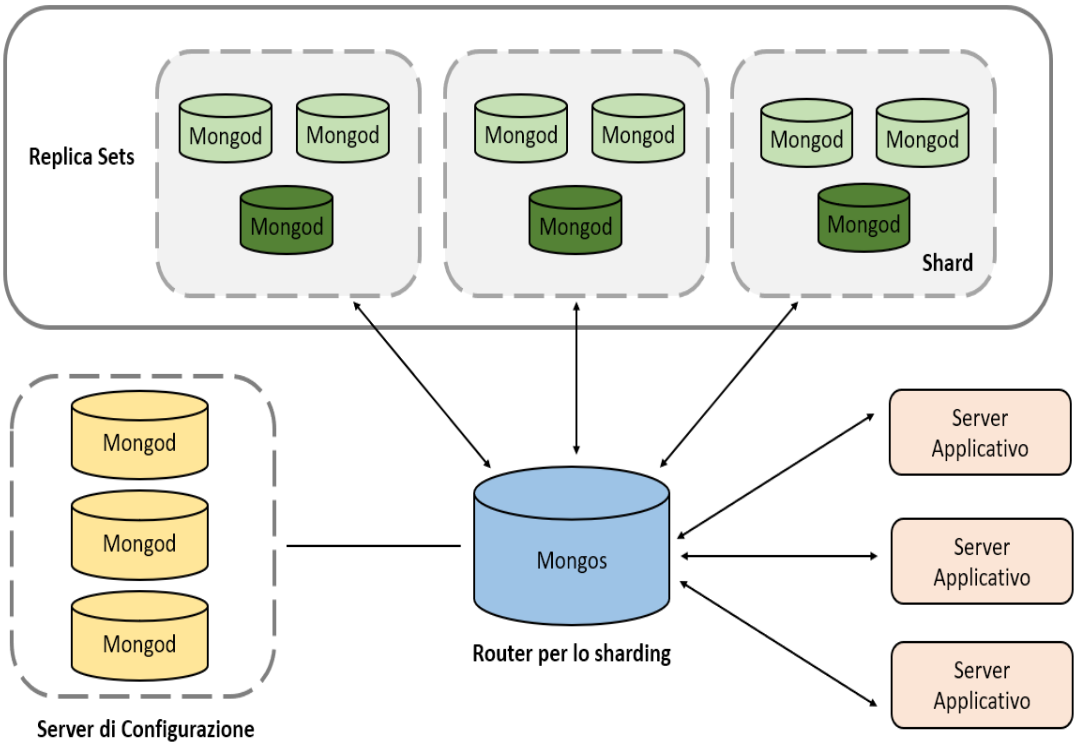

Figura 5.7: Architettura per lo sharding in MongoDB

- Supporto per le transazioni ACID (anche multi-documento): è previsto il concetto di sessione all'interno della quale sono possibili le operazioni inizio transazione e di commit. L'Atomicità permette che tutte le operazioni di una transazione siano annullate o completate come se fossero un'unica entità. La Coerenza o Consistenza obbliga il sistema a evolvere sempre da uno stato valido a uno stato valido successivo. L'isolamento garantisce che più transazioni possano essere eseguite contemporaneamente, senza che nessuna di queste possa visualizzare i risultati parziali delle altre. L'esecuzione di più transazioni contemporaneamente determina lo stesso risultato finale dell'esecuzione sequenziale. Infine, la Durabilità garantisce la persistenza di una transazione, anche in caso di errore del sistema.
  Le transazioni multi-documento di MongoDB sono previste solo per i server che utilizzano il motore di archiviazione WiredTiger [27] e attualmente sono limitate a un singolo set di repliche. Un altro limite di questa tipologia è l'impossibilità di influire sul catalogo del database (metadati associati al contenuto della base dati) e non possono eseguire operazioni che rientrino a far parte del dominio CRUD.

- Supporto per trasformazioni all'interno del database per la conversione dei tipi di dato: è prevista la conversione per dati di tipo double, string, objectId, boolean, date, interi, long e decimal usando la pipeline di aggregazione. Questo permette una trasformazione dei dati direttamente all'interno del database evitando di essere dipendenti da processi ETL (Extract, Trasform and Load).

- Costruzione su GUI (Graphical User Interface) di sofisticate pipeline di aggregazione ed elaborazione dati: l'Aggregation Pipeline Builder in MongoDB Compass [28] permette di sottomettere i documenti di una raccolta a più fasi di elaborazione, ogni fase e risultato parziale della pipeline può essere modificato in base alle esigenze applicative. L'utente, una volta creata la pipeline desiderata, può esportarne il codice e adattarlo a determinate operazioni da eseguire sulla propria applicazione.
- Ridotti i tempi di migrazioni per sistemi distribuiti;
- Autenticazione SHA-2;
- Cluster globali Atlas;
- Scalabilità multi-regione e sicurezza enterprise per MongoDB Atlas;
- MongoDB Stich e MongoDB Mobile.

5.4 Swagger

Durante la fase di sviluppo spesso si cerca di parallelizzare il lavoro degli sviluppatori ed è per questo motivo che generalmente si creano due gruppi di lavoro: uno dedicato allo sviluppo del backend e uno dedicato allo sviluppo del frontend.
Un aspetto importantissimo, volto alla comunicazione di informazioni utili sia tra persone appartenenti allo stesso team operativo sia a persone appartenenti a team differenti, è relativo alla creazione di una buona documentazione. In particolare il team di sviluppo dell'applicativo client deve usufruire spesso la documentazione relativa alle API.
Le applicazioni web a pagina singola interagiscono con il servizio per mezzo delle API REST, le quali si occupano dell'approvvigionamento di dati da e verso i server. Nonostante il team frontend possa lavorare per un certo periodo con una implementazione fittizia volta ad emulare il vero servizio (mock), è necessario fornirgli fin da subito un ottimo supporto alla documentazione del servizio sviluppato.
Per ogni possibile invocazione ad un servizio REST, è necessario conoscere diverse informazioni:

- l'URL da contattare comprensivo di porta TCP;
- il metodo HTTP per effettuare la richiesta;
- una descrizione di ogni azione che può essere svolta;
- i parametri in input includendo numero, tipologia e modalità di inoltro (sia che essi appartengano alla stringa query sia che essi appartengano al corpo della richiesta);
- i risultati da attendersi in funzione dell'esito dell'operazione sia come codice di stato HTTP sia come parametri restituiti.

A tal proposito è stato utilizzato Swagger [29]: un progetto open source totalmente dedicato alla documentazione delle API REST.
Può essere configurato con diversi framework, nel caso di questo progetto Django è stato utilissimo per la sua integrazione: in particolare è stata utilizzata la libreria "drf_yasg" la quale permette di generare automaticamente gli schemi per le API basandosi su Swagger/OpenAPI 2.0.
Swagger consiste in un file di testo (sia in formato YAML sia in formato JSON) dove sono descritte tutte le funzionalità di un'applicazione web e i dettagli di input e output in un formato studiato per essere interpretabile correttamente sia dagli umani che dal software.
I vantaggi principali sono una chiara spiegazione delle funzionalità di un'applicazione, oltre che la possibilità di automatizzare la generazione di codice client/server sfruttando direttamente i vincoli definiti nello schema.
Oltre al supporto per la creazione della documentazione, fornisce altri utili strumenti per la creazione di un modello per le API conforme ai migliori standard seguiti dagli sviluppatori. Un altro vantaggiosa tecnologia è Swagger Inspector, il quale permette la possibilità di testare le API sviluppate. Swagger fornisce un supporto base per testare ogni API e questo lo rende utile non solo per gli sviluppatori frontend ma anche per quelli backend.

Capitolo 6

Autenticazione

Per lo sviluppo del processo di autenticazione sono state seguite specifiche precise predisposte dalle parti interessate. Sulla base dei requisiti sono stati modellate due diverse tipologie di autenticazione, una per il LAS e una per la Biobanca, entrambe fondanti su alcune tecnologie comuni che verranno brevemente descritte in seguito.

6.1 Tecnologie

Il progetto allo stato attuale rappresenta un prototipo e quindi non è ancora stato messo in produzione. In particolare per quanto riguarda gli aspetti di sicurezza sono state considerate tecnologie e fatte assunzioni che allo stato dell'arte attuale sono ancora modificabili o reversibili. Lo scopo di questa prima implementazione è stata, infatti, quella di poter fornire una base funzionale che possa essere completata, messa in sicurezza e mantenuta dal team di sviluppatori incaricato.

6.1.1 HTTPS

L'HyperText Transfer Protocol over Secure Socket Layer (HTTPS) [30] è un protocollo utilizzato per instaurare una comunicazione sicura attraverso una rete di computer utilizzato su Internet. Si basa sul protocollo HTTP (HyperText Transfer Protocol) in combinazione con il protocollo Transport Layer Security (TLS).
Il motivo per cui è stato utilizzato è quello di poter fornire un'autenticazione dell'applicativo in uso, fornendo la proprietà di integrità dei dati scambiati tra le parti comunicanti. Nel suo uso più comune, HTTPS fornisce quindi l'autenticazione del server web e del sito web associato con cui una delle parti sta comunicando.
In generale protegge la comunicazione da attacchi di tipo MITM (Man in the Middle) [31]. Fornisce, attraverso una cifratura bidirezionale della comunicazione instaurata, un supporto alla garanzia della privacy della comunicazione evitando le possibili operazioni di "eavesdropping" (intercettazione della conversazione senza il consenso delle parti) e di " tampering" (manomissione e quindi falsificazione del contenuto della comunicazione) [32].

La cifratura del protocollo HTTP include:

- l'URL;
- i parametri di query;
- le intestazioni della connessione;
- i cookies;

La sicurezza di HTTPS è garantita dal protocollo TLS in combinazione con l'utilizzo di certificati definiti dallo standard X.509 [33], utilizzati per autenticare il server ed eventualmente anche il client. L'apparato delle autorità certificative è necessario al fine di verificare il rapporto che sussiste tra il certificato e il suo proprietario, oltre a generare la firma e gestire la validità dei certificati ed eventuali revoche.
L'utilizzo del protocollo HTTPS deve essere esclusivo, quindi viene vietato l'uso di HTTP per evitare di esporre gli utenti a determinati attacchi.

6.1.2 TLS

Transport Layer Security (TLS) [32] è un protocollo crittografico che permette di instaurare un canale sicuro durante una connessione su reti TCP/IP fornendo autenticazione, integrità dei dati e confidenzialità operando al di sopra del livello di trasporto.
È un protocollo standard IETF la cui ultima versione è definita nella RFC 5246.
In generale l'autenticazione mediante TLS è unilaterale: solamente il server si autentica, mentre il client rimane anonimo e non autenticato verso il server.
Nella pratica l'autenticazione del server è necessaria per il software di navigazione e per l'utente. Il browser è in grado di validare il certificato del server controllando che la firma digitale dei certificati del server sia valida e riconosciuta da una autorità di certificazione. Occorre anche analizzare il contenuto del certificato rilasciato e controllarne la catena di certificazione per essere sicuri della veridicità dell'autenticazione.
Il protocollo TLS permette anche un'autenticazione bilaterale, chiamata anche mutua autenticazione. Solitamente il suo utilizzo è destinato ad applicazioni aziendali, in cui entrambe le parti si autenticano in modo sicuro scambiandosi i relativi certificati. Questa tipologia di autenticazione richiede quindi che anche il client possieda un proprio certificato digitale, questo scenario potrebbe essere considerato in futuro durante l'analisi di sicurezza dell'applicativo sviluppato.

6.1.3 JWT

Una delle funzionalità più ricercate e indispensabili in una moderna applicazione web è quella di poter determinare chi può accedere a un set di determinate pagine, creando profili utente con diverso livello di autorizzazione e profili amministratore. Questo meccanismo è ricercato anche per filtrare opportunamente le API che espone un server web.
Un servizio REST, rispettando la comunicazione nativa del protocollo HTTP, è per definizione privo di stato. L'unica gestione dello stato, quando strettamente

necessaria, avviene sul client ottimizzando le prestazioni del server, il quale non deve curarsi di cambiamenti nello stato del sistema. Questo approccio rende quindi inappropriato l'utilizzo di cookie di sessione per il meccanismo dell'autenticazione.

Una prima soluzione molto semplice potrebbe essere quella di effettuare un'autenticazione di tipo "Basic", nella quale ad ogni richiesta il client passa al server le credenziali in modo da permettere la sua autenticazione. Questo approccio tuttavia porta ad un overhead lato backend: basti immaginare alle query effettuate ad ogni richiesta per effettuare il controllo delle credenziali.
Una soluzione in armonia con il principio "stateless", scalabile e senza alcun overhead lato server è rappresentata dal JWT (JSON Web Token) [34].

JWT è un token che agisce come un contenitore per le "affermazioni" degli utenti. Le affermazioni sono informazioni che l'utente fornisce di se stesso come per esempio dati di contatto, di privilegio, di personalizzazione del profilo e sicuramente ultima ma non meno importante la scadenza del token stesso. L'intero contenuto del JWT è firmato lato server secondo la specifica JSON Web Signature (JWS), in questo modo il backend è in grado di riconoscere se il token arrivato da una richiesta di un determinato client è stato generato da lui in un momento precedente oppure se è stato manomesso. Di seguito è riportato un esempio di JWT sia codificato sia decodificato:

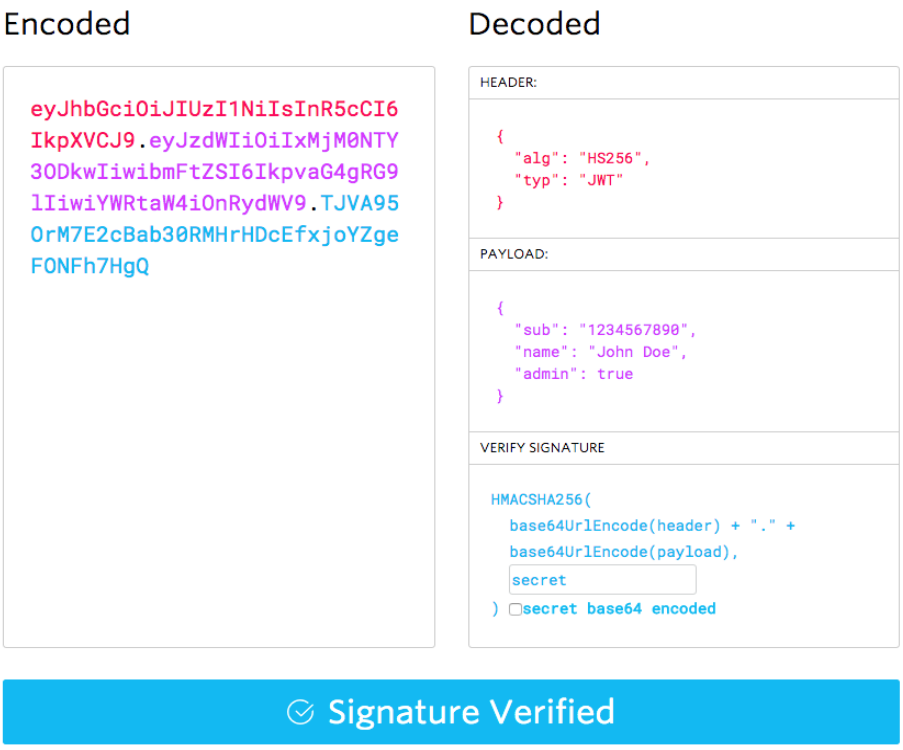

Figura 6.1: JSON Web Token

Come è mostrato in Figura 6.1, il JWT è formato da tre componenti principali:

- Intestazione: viene indicato il tipo di token e l'algoritmo di cifratura utilizzato per la firma.

- Corpo: contiene i dati personalizzabili dell'utente, il suo scopo è quello di far viaggiare informazioni dal server al client, non il viceversa in quanto il server dovrà sempre controllare che il token corrisponsa a quello generato in precedenza.
- Firma: è il risultato di una funzione di hash che prende in input la codifica base64 dell'intestazione concatenata con il carattere punto alla codifica base64 del corpo, il tutto crittografato con una chiave segreta.

L'utilità di base di questa tecnologia è quella di poter permettere lo scambio di informazioni legate all'autenticazione di un utente. Attraverso la firma il backend è in grado di controllare se il token è genuino o se è stato manomesso da malintenzionati, permettendo così solo ad utenti autenticati di effettuare richieste che verranno prese a carico dal server.
Riassumendo i passi principali del processo di autenticazione tramite JWT (sotto le precedenti ipotesi di HTTPS + TLS) sono:

1. Tramite la pagina di login l'utente inserisce le credenziali di accesso (username e password), quindi l'applicativo client invia tramite metodo HTTP POST una richiesta di autenticazione all'omonimo server;
2. Il server di autenticazione controlla la correttezza delle credenziali di accesso e in caso positivo genera un JWT contenente le informazioni dell'utente e una scadenza, questo token viene firmato e autenticato con la chiave segreta del server;
3. A questo punto il client è autenticato e l'utente ha accesso al contenuto dell'applicazione, ad ogni richiesta successiva il JWT verrà inviato al server applicativo incapsulandolo in una opportuna intestazione;
4. Il server applicativo, ad ogni richiesta successiva del client contenente un JWT, controllerà la firma e l'autenticazione e successivamente anche la validità temporale.

Questa tecnologia ha alcuni vantaggi: permette di superare il problema del segreto condiviso, infatti solo il server deve conoscere la chiave privata e offre una buona scalabilità in quanto i dati dell'utente sono già presenti nel token e questo evita di dover effettuare delle interrogazioni al database per recuperare lo stato della sessione.

In questo scenario è utile implementare anche un meccanismo di rinegoziazione del JWT. Nel progetto la finestra temporale del token, mostrata in Figura 6.2, è stata fissata ad un certo intervallo, mentre la rinegoziazione è possibile solo in un sotto intervallo molto vicino alla scadenza. La rinegoziazione è automatizzata in questo piccolo intervallo non appena il client effettua una richiesta al server. In caso invece di una forte inattività dell'utente tale processo di rinegoziazione non avverrà e così l'utente risulterà nuovamente non autenticato.
Questo approccio dona una certa continuità operativa ricercata soprattutto durante il lavoro quotidiano di un operatore che accede ai dati del LAS o della Biobanca.
Per evitare che le richieste lato client effettuate in un momento di forte inattività

non vadano perdute, si ricorda che esiste il meccanismo di cache che permette di salvare in locale ugualmente le modifiche ai dati sui quali si stava operando e, in caso di una richiesta fallita per non attività, sarà possibile autenticarsi nuovamente, quindi ripristinare il contenuto di una vista per mezzo dei dati salvati in precedenza nell'Indexed DB.

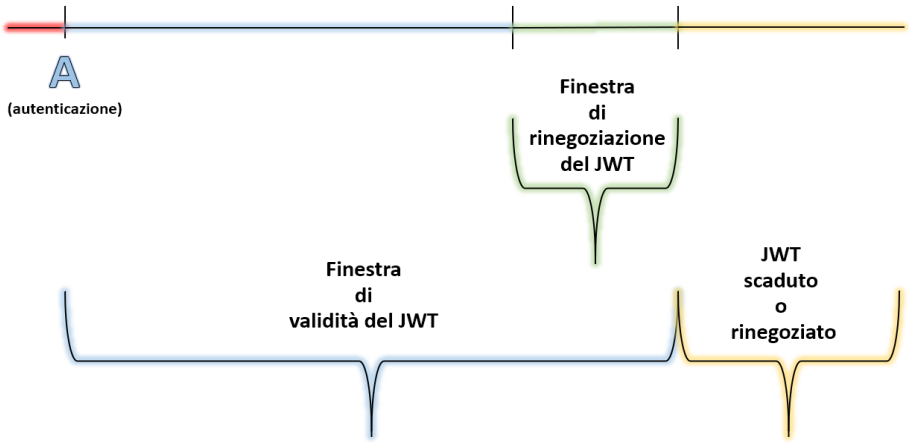

Figura 6.2: Finestra temporale del JWT

Il JWT tuttavia ha alcuni svantaggi che devono essere presi in considerazione:

- Utilizzare una chiave segreta unica per firmare il token rappresenta un "single point of failure" dal punto di vista della sicurezza: se la chiave viene scoperta l'intero sistema è compromesso. Una soluzione potrebbe essere quella di non utilizzare sempre la stessa chiave, cambiandola per esempio ogni giorno. Questo approccio è utilizzato in sistemi fortemente sensibili come per esempio può essere una Banca;

- Se le informazioni contenute nel token aumentassero potrebbero causare un overhead;

- I client non sono gestibili dal server e quindi non esiste un vero e proprio logout lato backend. Lato client il JWT viene pulito e dimenticato dal browser. In caso di compromissione di uno dei client l'unica alternativa ad una buona gestione dei JWT è quella di aggiungerlo ad una apposita blacklist per rifiutarne le richieste. Usando questa soluzione tuttavia ci allontaneremmo dal costruire un servizio privo di stato.

- Il server non è in grado di distinguere gli utenti loggati e quelli non. Per esempio immaginiamo il caso in cui un utente effettui il login: verrà emesso un jwt con validità entro una certa finestra temporale. Ora se l'utente effettua il logout, l'applicativo client cancellerà localmente il jwt emesso in precedenza senza comunicare nulla al server, il quale quindi non tiene di fatto traccia di chi è loggato. Quando un utente chiede di autenticarsi il server non sa se si tratta di un utente già loggato o meno;

6.1.4 Storage del JWT

In generale il JWT deve essere salvato lato client per poterlo inviare ad ogni richiesta al server. Le alternative principali sono utilizzare lo storage del browser oppure i cookie. Entrambe le soluzioni comportano vantaggi e svantaggi, attualmente una delle due soluzioni è stata adottata ma questo non comporta che non possa essere deciso in futuro di adottare la complementare o una nuova strategia eventualmente. Sulla base della tecnica adottata dovranno essere utilizzate misure cautelari e impiegate strategie diverse per mettere il sistema in sicurezza.
Successivamente è riportata una breve analisi.

Local Storage o Session Storage

Questo primo approccio permette di accedere al contenuto del JWT direttamente dal codice sorgente del client. Ne comporta la possibilità di estrarre le informazioni contenute nel corpo del token andando a personalizzare, sulla base di queste, il contenuto visualizzato dall'utente finale. Questo comporta una buona flessibilità e permette di sfruttare le potenzialità del JWT, tuttavia espone ad una tipologia di attacco: XSS (Cross Site Scripting) .

XSS [35] è una vulnerabilità di sicurezza tipica delle applicazioni Web. In questo scenario un utente malintenzionato invia uno script alla vittima mediante l'iniezione dello script dannoso in un sito Web attendibile. La vittima fidandosi dell'attendibilità del sito Web, apre una finestra che eseguirà sul browser lo script malevolo. L'esecuzione di script dannosi può, oltre a cambiare l'albero DOM, accedere al contenuto memorizzato sul browser.
In questo contesto il framework Angular considera, attraverso l'utilizzo dell'API DOMSanitizer [36], tutti i dati come non attendibili. Pertanto, per impostazione predefinita, effettua la loro sanificazione. Essenzialmente ogni tag HTML trovato nei dati applicativi è identificato e ripulito.
Content Security Policy (CSP) [37] è un ulteriore livello di sicurezza che aiuta a rilevare e mitigare alcuni tipi di attacchi, tra cui Cross Site Scripting. Per abilitare CSP bisogna configurare il server Web, sviluppato con Django framework, in modo che restituisca un'intestazione HTTP di tipo Content-Security-Policy.

Utilizzando il Session Storage, la visibilità delle coppie chiave-valore è relativa alla finestra del browser in cui sono state generate. Una volta chiusa la finestra, i dati contenuti nell'istanza dello storage corrispondente verranno distrutti assieme alla finestra stessa.
Questa limitazione non è prevista con l'utilizzo del Local Storage.

Cookie

L'altra soluzione è rappresentato dai cookie. Il server di autenticazione, in caso di ricezione di credenziali corrette da parte di un utente, invia una risposta la quale imposterà in automatico un cookie lato client contenente il JWT.
Il cookie settato dal server sul client possiederà i seguenti attributi:

- Secure: indica al browser che il cookie deve essere restituito all'applicazione solo tramite connessioni crittografate, quindi tramite HTTPS;
- HTTP Only: indica che il cookie non sarà in alcun modo accessibile all'applicativo client, questo permette di mitigare il problema dell'attacco XSS.
- SameSite: l'utilizzo dei cookie in generale espone l'intero sistema ad attacchi di tipo Cross-site request forgery (XSRF).

  > "CSRF (Cross-Site Request Forgery) [38] è un attacco che costringe un utente finale a eseguire azioni indesiderate su un'applicazione Web in cui è attualmente autenticato. Gli attacchi CSRF mirano in modo specifico alle richieste di modifica dello stato, non al furto di dati, poiché l'autore dell'attacco non ha modo di vedere la risposta corrispondente alla richiesta falsificata. Con un piccolo aiuto dell'ingegneria sociale (come l'invio di un collegamento via e-mail o tramite l'uso di chat), un utente malintenzionato può ingannare gli utenti di un'applicazione Web nell'esecuzione di azioni scelte dall'utente. Se la vittima è un utente normale, un attacco CSRF riuscito può costringere l'utente a eseguire richieste di modifica dello stato come per esempio il trasferimento di fondi, la modifica dell'indirizzo e-mail e così via. Se la vittima è un account amministrativo, CSRF può compromettere l'intera applicazione Web."

  Lato client il modulo Angular HttpClient fornisce un supporto integrato per eseguire controlli per questo generi di attacchi, tuttavia la sicurezza deve essere implementata lato server. Django framework dispone di alcuni plugin e moduli per far fronte a questa tipologia di attacco.
  Una buona pratica in questo contesto è l'utilizzo dell'attributo cookie SameSite, il quale fornisce protezione da CSRF con la possibilità di assegnare due diversi valori "strict" o "lax", in base alle necessità.

  > "SameSite [38] impedisce al browser di inviare questo cookie insieme a richieste cross-site. L'obiettivo principale è quello di mitigare il rischio di perdite di informazioni di cross-origin. Fornisce inoltre una certa protezione contro gli attacchi di contraffazione delle richieste cross-site. I valori possibili per questo attributo sono lax o strict."

Il valore strict limita l'utilizzo del cookie durante la navigazione dell'applicativo. Immaginiamo di accedere a un link riguardante la nostra applicazione Web tramite email o su un forum, sia che il collegamento sia regolare o malevolo, non sarà possibile utilizzare il cookie verso il server per accedere al contenuto.
Questo caso d'uso, per esempio, è riconducibile alla protezione delle pagine transazionali di una Banca e viene utilizzato per evitare collegamenti esterni.
Per quanto riguarda il valore lax, invece, viene fornita una maggiore usabilità in caso di collegamenti esterni, con un certo compromesso di sicurezza.

Una limitazione derivante dall'utilizzo dell'attributo SameSite è rappresentata dal fatto che non è ancora stato implementato in tutte le principali versioni del browser.

A partire da novembre 2017 è stato implementato in Chrome, Firefox e Opera.
Infine è da sottolineare il fatto che i cookie possono essere disabilitati, e questo comporterebbe un limite all'operatività di questa soluzione.

6.2 Processo di Registrazione

Il processo di registrazione è stato personalizzato sulla base delle specifiche. Sono state create tre tipologie di registrazione differenti, in particolare due di esse vengono utilizzate nell'ambito dell'applicativo client del Laboratory Assistant Suite e una per la Biobanca.
La necessità di dover disporre di un set di meccanismi differenti per potersi registrare è dovuto a causa della complessità delle relazioni che intercorrono tra gli utenti del LAS e dal fatto che in generale un utente che lo utilizza può accedere eventualmente al contenuto della Biobanca mentre non potrà mai accadere il viceversa.
Tutte le registrazioni si basano su principi comuni stabiliti nel documento delle specifiche:

- E' previsto un campo email obbligatorio per ogni utente;
- Ogni processo di registrazione è accessibile solo per account email che non sono ancora registrati in alcun modo;
- La registrazione avviene senza alcuna immissione di password;
- E' previsto un reCAPTCHA di Google. Si tratta del frutto di un progetto sviluppato da alcuni ricercatori della Carnegie Mellon University di Pittsburgh, in Pennsylvania, e acquistato nel 2009 da Google. Sostanzialmente sono test semplici che servono a distinguere gli umani dai robot.
- Ogni registrazione effettuata da un utente rimane pendente in attesa di validazione da parte del sistema. Questo risulta immediato e automatico per il vasto pubblico della Biobanca mentre richiede un'accurata analisi ed eventuale approvazione manuale da parte dell'Amministratore per quanto riguarda il LAS;
- Quando una richiesta di registrazione viene accettata il sistema invia all'account dell'utente un'email di conferma con username e password provvisoria, una volta effettuato il login l'utente potrà cambiare il proprio segreto utilizzato per l'accesso con una nuova password definitiva (durante questa fase l'utente è considerato dal server ancora come pendente, e se non completerà l'operazione di cambio password non potrà accedere alle funzionalità del sistema).

6.2.1 Registrazione Biobanca

Nell'ambito della Biobanca l'utente inserisce i seguenti campi obbligatori:

1. Nome

2. Cognome

3. Email

Gli utenti della Biobanca vengono considerati di tipo "GUEST" mentre un account amministratore è considerato di tipo "ADMIN" (per entrambi i frontend). La distinzione dei ruoli in questo contesto è molto semplice e se ne lascia il compito di descriverli nel Capitolo 7.

6.2.2 Registrazione LAS

Il LAS rappresenta una realtà molto più complessa rispetto alla Biobanca:

- La rete operativa del LAS è costituita da diversi progetti.
- Ogni progetto di lavoro è assegnato ad uno o più Working Group.
- Ogni utente per iscriversi deve quindi dichiarare di lavorare per un determinato Working Group.
- Gli utenti possono essere suddivisi in tre categorie:
  - PI: è il responsabile legale e di gestione del Working Group, rappresenta il grado massimo in un progetto di lavoro (tutti gli utenti possono diventare PI di un WG se necessario);
  - Vice PI: è un utente che può assumere il ruolo futuro di PI, possiede le funzionalità di gestione del PI;
  - Semplice: è un utente che lavora presso un Working Group senza avere responsabilità di gestione del gruppo di lavoro.
- Gli utenti dopo la fase preliminare di registrazione possono lavorare o diventare responsabili per Working Group diversi (effettuando una opportuna richiesta di adesione per un determinato gruppo di lavoro o di creazione di uno nuovo). Riassumendo i WG rappresentano sostanzialmente dei gruppi sociali di utenti mentre i Progetti di Ricerca aggregano utenti (appartenenti a determinati WG) definendo regole che influiscono sulle funzionalità che sono applicabili ai campioni biologici.

Sulla base di queste relazioni che intercorrono tra utenti e gruppi di lavoro sono state modellate due tipologie di registrazione che permettono di accedere al sistema sul concetto di garanzia o fiducia.

Registrazione del Working Group

La prima registrazione possibile è quella del Working Group. In questa fase vengono immessi i seguenti dati:

- Nome del Working Group
- Istituto di appartenenza
- Dati del PI:
  - Nome
  - Cognome
  - Email
  - Numero di telefono
- Dati dei membri, per ognuno (nota: questo insieme può essere vuoto):
  - Nome
  - Cognome
  - Email
  - Indicazione di Vice PI (attributo vero o falso)

Questa procedura permette di snellire l'iter di registrazione di un nuovo WG, poiché raggruppa in un solo processo la definizione dei suoi utenti evitando che ognuno di essi spenda del tempo per ricercare il gruppo corretto per registrarsi ed evitando numerosi errori umani portati dall'operazione di ogni singolo.
Per migliorare l'esperienza utente è stato inserito, come mostrato in Figura 6.3, uno stepper che permette di effettuare una registrazione "guidata" del Working Group.

Registrazione del singolo utente

Una volta che un Working Group è stato registrato nasce la necessità di aggiungere nuovi membri nel tempo. Per questo motivo è stata modellata una registrazione per il singolo utente che vuole aderire ad un gruppo di lavoro.
In questa fase vengono immessi i seguenti dati:

- Dati relativi all'utente:
  - Nome
  - Cognome
  - Email
- Nome del Working Group (bisogna scegliere una checkbox da una lista che appare successivamente alla ricerca dei gruppi di lavoro basata sul Cognome del PI)

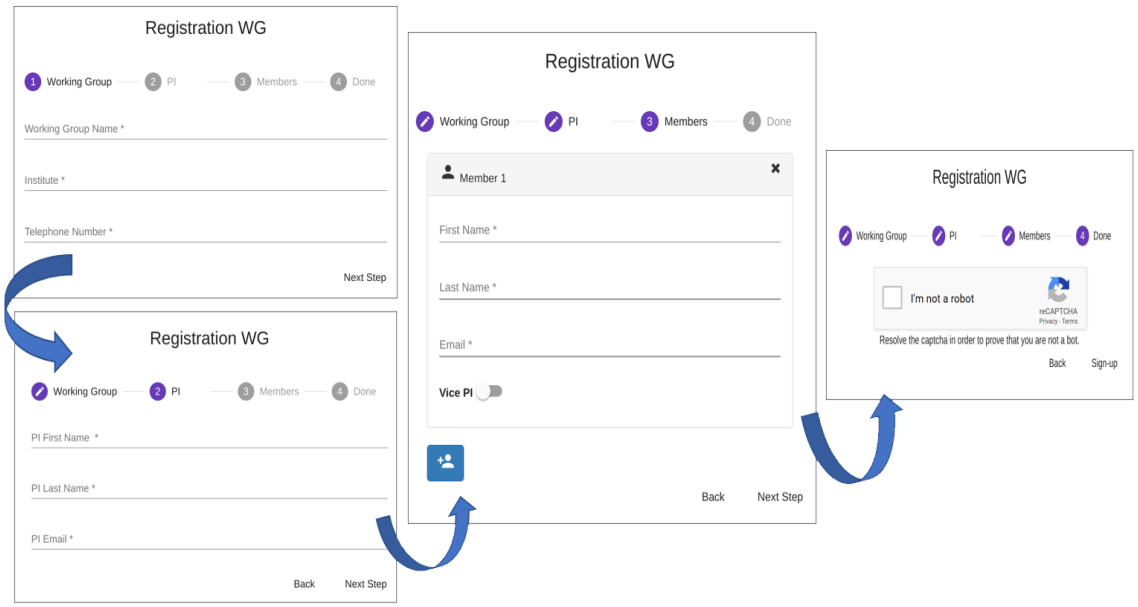

Figura 6.3: Passaggi per la registrazione di un WG

Attraverso questo modulo di registrazione l'utente è vincolato a poter scegliere solamente un Working Group. Questa restrizione è pensata per facilitare il processo di approvazione da parte dell'Amministratore, il quale potrà fare riferimento all'unico garante (PI) del Working Group selezionato.
Una volta che un utente accede al sistema del LAS può tranquillamente richiedere di accedere ad altri Working Group in un secondo momento. Questa tipologia di richieste vengono modellate come richieste di privilegio, e vanno sempre valutate.

Il ruolo del PI di fatto è pensato per poter decentralizzare la responsabilità dell'Amministratore di accettare o meno dei membri nel proprio Working Group, sostituendosi lui stesso al processo di autorizzazione e gestione dei permessi di accesso al proprio gruppo di lavoro.

Questi due moduli di registrazione fanno emergere, inoltre, un nuovo caso d'uso da prendere in considerazione:

1. Il PI crea un Working Group senza alcun membro;
2. Il PI viene validato dall'Amministratore e di conseguenza il WG creato;
3. Gli utenti che vogliono aderire al gruppo effettuano la richiesta tramite il modulo di registrazione singola;
4. Il PI accetta o rifiuta la loro adesione.

6.3 Login

Per quanto riguarda il processo di Login, come mostrato in Figura 6.4, è stato previsto un form molto semplice che prevede:

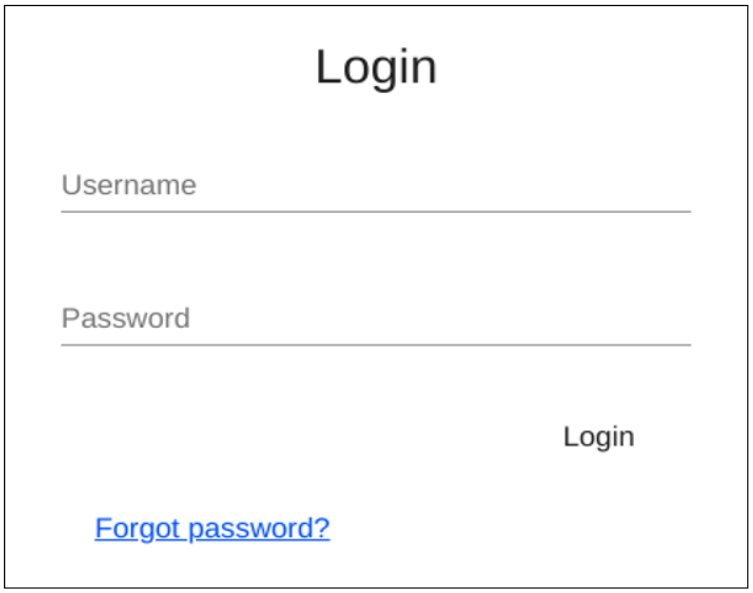

Figura 6.4: Form di Login

- Username
- Password
- Bottone per autenticarsi con le credenziali inserite
- Link alla sezione cambio password

6.3.1 Reset Passwrod

Come anticipato in precedenza è stato previsto un modulo dedicato alla gestione del cambio password degli utenti. In particolare è possibile:

- se l'utente non è autenticato, può accedere tramite il link presente nel form di login a una pagina dove è possibile inserire il proprio indirizzo email associato all'account e richiedere di ricevere un'email di reset come mostrato nella Figura 6.5. Questa email contiene un collegamento, costruito con un hash code generato dal server, il quale permette di accedere ad un form di reset password. In particolare in quest'ultimo, Figura 6.6, bisogna inserire la nuova password (due volte per confermarla) e viene valutata con una barra a gradazione cromatica la forza di tale password; è possibile inoltre utilizzare un apposito bottone per generare la nuova password in automatico.
- se l'utente risulta autenticato, può accedere a un form di cambio password molto simile al precedente con l'unica differenza di dover inserire anche la vecchia password per confermare la procedura.

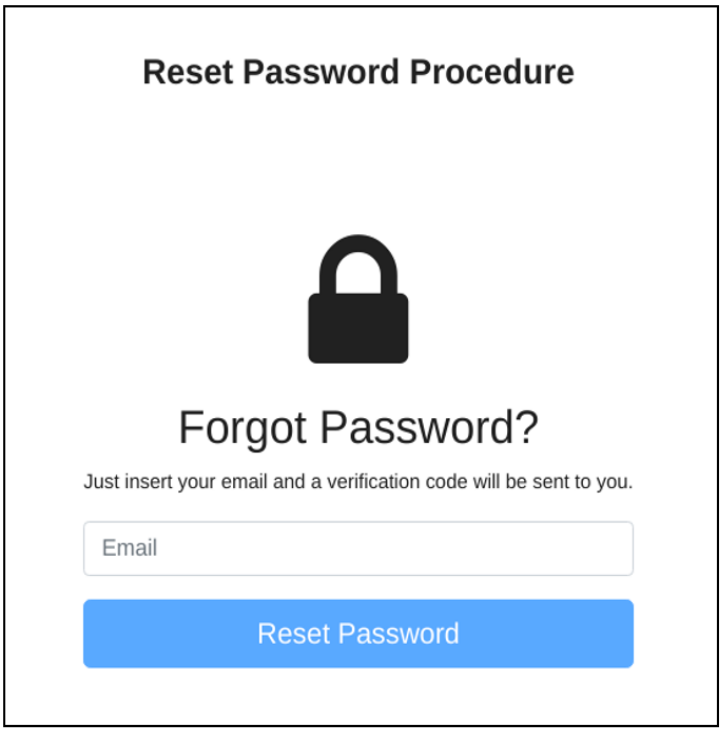

Figura 6.5: Form di Richiesta Reset

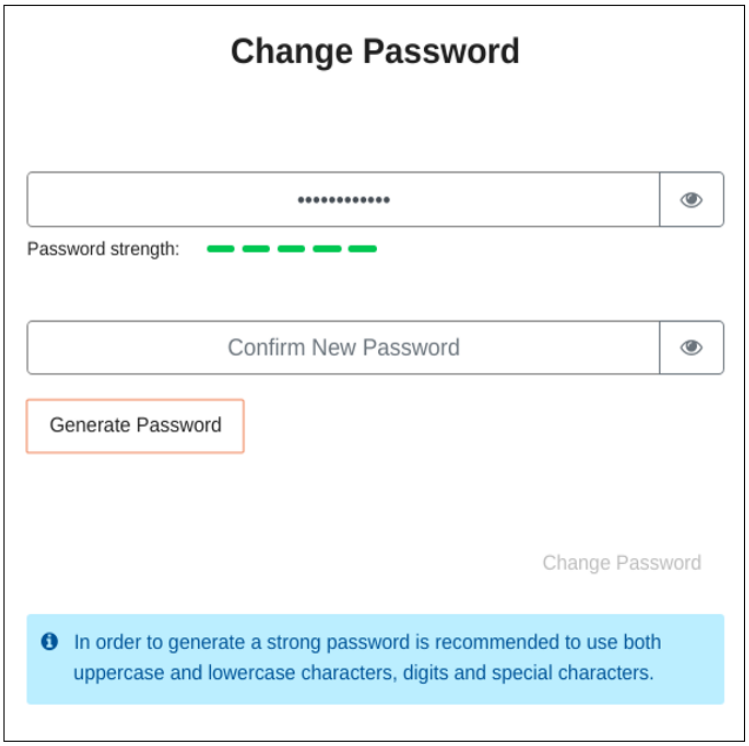

Figura 6.6: Form di Reset Password

Capitolo 7

Autorizzazione

7.1 Gestione dei Privilegi

La gestione dei privilegi avviene in maniera centralizzata da parte del framework, il quale ha il compito di coordinare le attività degli utenti che usufruiscono delle molteplici interfacce client. La gestione dei privilegi si basa sui seguenti concetti principali:

1. Permesso legato al dato: ogni dato reso persistente nella base dati viene associato, secondo un modello particolare, a un insieme di permessi e funzionalità. Questo permesso è utilizzato come un filtro durante una richiesta da parte di un utente. L'utente deve sempre specificare nella richiesta il Contesto di utilizzo del dato.

2. Contesto di utilizzo del dato: rappresenta la vista per la quale si sta richiedendo il dato. E' un parametro fondamentale che identifica in quale ambito è richiesto il dato. Rappresenta quindi un ulteriore filtro.

3. Funzionalità legata al rendering della vista: lato client vengono definite delle funzionalità che permettono ad un utente che le possiede di poter accedere a un insieme di viste specifico.

7.1.1 Permessi associati al dato

La gestione dei permessi ha una complessità non trascurabile. Nel framework sviluppato il permesso viene associato al dato, il quale si porta con sé delle etichette che permettono di riconoscere chi può utilizzarlo e in quale modo.
Il modello che è stato sviluppato sulla base delle necessità operative (Figura 7.1), è formato dalle seguenti entità principali:

- Funzionalità: definisce l'accesso ad una classe di funzionalità del frontend. Viene associato agli USER per una particolare Etichetta.

- Utente: è un entità che identifica l' utente. Ogni Utente possiede un attributo "concesso", ovvero una matrice di Funzionalità x Etichette, i cui valori determinano l'accesso alle schermate (Funzionalità) e ai dati visualizzabili in base al Contesto. La permissività è modellata con due valori possibili (accesso consentito oppure accesso non consentito).

- Etichetta: rappresenta una classe astratta che descrive con l'attributo "utenti" una collezione di persone che possono interagire con l'etichetta.

  – WG: sottotipo di Etichetta che rappresenta il gruppo di lavoro. Ogni Utente durante la fase operativa contrassegna i dati generati con questa Etichetta di riferimento, quindi se un Utente appartiene a più gruppi di lavoro, quando agisce su un dato lo fa nel Contesto di uno specifico WG.

  – Progetto: sottotipo di Etichetta che modella un progetto per la collezione di una Bio-entità. L'attributo "operazioni" descrive le funzionalità ammesse per i dati contrassegnati con questo tipo di Etichetta (in base anche al consenso informatico). In questo modo è possibile definire l'accesso ai dati in maniera inclusiva (i.e., whitelist) e in maniera esclusiva (i.e., blacklist).

- Bio-entità: è un tipo astratto. Diverse entità potranno estendere questa tipologia di dato. Possiede l'attributo complesso "regole di accesso" che determina per quali Etichette il dato è disponibile in lettura o scrittura. E' caratterizzata anche da un attributo chiamato "propagazione regole di accesso" che descrive come è composto per le bio-entità figlie l'attributo "regole di accesso" in maniera di default.

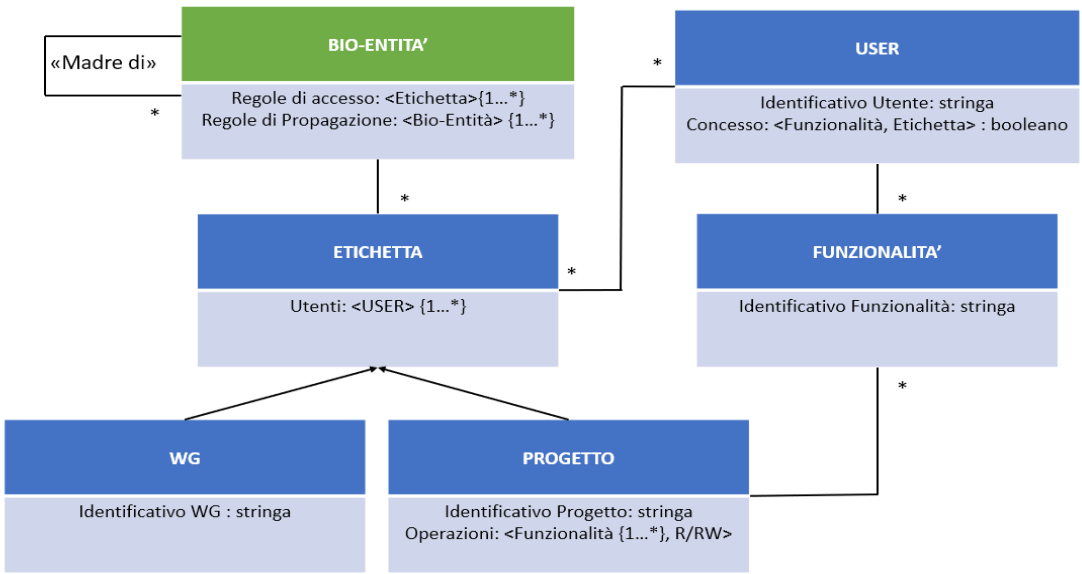

Figura 7.1: Modello Permessi

Per la gestione dei permessi associati ai dati viene utilizzato il meccanismo delle liste. Esistono quindi una whitelist e una blacklist. La prima indica chi può accedere

mentre la seconda indica solamente chi non può. In questo scenario risulta molto più sicura e meno permissiva l'impiego della prima, tuttavia anche la seconda variante è stata inclusa nella modellazione del progetto per avere uno strumento che possa essere utile in determinate situazioni.

In seguito è riportato, per completezza, un esempio della matrice di Funzionalità x Etichette di un determinato Utente applicata nel caso di whitelist (il valore 1 indica la concessione del permesso, il valore 0 indica la non concessione del permesso):

	Etichetta WG1	Etichetta Progetto A	Etichetta Progetto B
Funzionalità X	1	0	1
Funzionalità Y	1	1	0
Funzionalità Z	1	1	1

Tabella 7.1: Matrice Funzionalità x Etichette

L'utente può accedere alle funzionalità X, Y e Z. Tuttavia i dati che potrà usare al loro interno variano a seconda dei valori di tale matrice.
Per esempio, per la Funzionalità di tipo X l'utente potrà interagire solamente con i dati che rispettano entrambi i due seguenti requisiti:

- hanno entrambe o almeno una delle etichette WG1 e Progetto B;
- non hanno l'etichetta Progetto A.

Quindi l'utente che possiede quella determinata matrice potrà accedere alla Funzionalità X per dati etichettati come [WG1, Progetto C] ma non vi potrà accedere per dati etichettati come [WG1, Progetto A].

7.1.2 Contesto

Rappresenta un filtro ulteriore sui dati. Ogni qualvolta un utente effettua una richiesta, il contesto viene specificato nel corpo della richiesta. In questo modo è possibile conoscere la vista per la quale si stanno fornendo i dati.

7.1.3 Rendering frontend sulla base delle funzionalità

Le applicazioni web a pagina singola permettono l'esecuzione del codice client direttamente sulla macchina dell'utente. In questo modo il contenuto delle pagine web è già presente durante il primo avvio dell'applicazione. Il frontend attraverso un insieme di servizi richiede tramite il protocollo HTTP al Servizio REST solamente i dati necessari a popolare le viste.
Questa caratteristica rende sicuramente l'applicativo molto più "responsive" e veloce, tuttavia fa nascere il problema di dover gestire con l'utilizzo di appositi guardiani, il rendering delle pagine.
Immaginiamo di essere un utente base con il livello minimo di privilegio, sulla nostra macchina verrà scaricato anche il codice e quindi le viste relative al profilo

amministratore. Il frontend deve essere in grado di riconoscere, durante il processo di autenticazione, quali sono le funzionalità associate ad un utente in modo da visualizzare solo un sotto-insieme coerente di viste.

Il JWT si presta un'ottima tecnologia per permettere di inserire nel suo corpo le funzionalità associate ad un utente per lo svolgimento delle sue attività: il server all'atto della creazione del token indica quali funzionalità sono previste per quell'utente, in caso un malintenzionato cercasse di modificare i suoi privilegi, modificando di conseguenza il JWT, il backend, basandosi sulla firma, sarebbe in grado comunque di determinare tali modifiche e invalidare la richiesta proveniente dal client.

Nello specifico di Angular Framework è stata utilizzata la libreria "ngx-permissions". Questa si adatta perfettamente alle esigenze descritte: durante il caricamento del primo modulo, successivamente alla fase di login, viene esaminato il contenuto del JWT e quindi caricate tutte le funzionalità elencate nel suo corpo. Le URL e il contenuto delle viste possono essere posti sotto il controllo di un guardiano implementato dalla libreria. Il guardiano altro non è che un servizio che controlla se l'utente ha il permesso per accedere o meno.
Bisogna sottolineare che l'utilizzo di questa libreria non fornisce in alcun modo una garanzia di sicurezza. Infatti, la gestione vera e propria dei permessi legati ai dati e alle collezioni avviene lato server come affermato in precedenza. Basti pensare ad un utente che modifichi il proprio JWT affermando di avere tutte le funzionalità esistenti: in quel caso potrebbe avere accesso a tutte le viste sul client create dal programmatore, tuttavia, anche il quel caso, qualsiasi richiesta verso il Servizio REST sarebbe invalidata in quanto verrebbe riconosciuta la manomissione del token, e quindi il massimo risultato ottenibile dall'utente sarebbe di poter visualizzare delle viste prive di contenuto.
Quindi in conclusione: l'utilizzo di questo meccanismo lato client non vuole sostituirsi ad una gestione dei privilegi ma solo consentire un utilizzo corretto dell'applicativo, mostrando un contenuto coerente con le funzionalità dell'utente migliorando la sua esperienza.

In seguito è riportato, per completezza, un esempio molto semplice di una matrice Funzionalità x Utente (il valore 1 indica che l'utente possiede la funzionalità, il valore 0 indica che l'utente non possiede la funzionalità):

	Utente A	Utente B	Utente C
Funzionalità ADMIN	1	0	0
Funzionalità BIOBANCA	1	1	1
Funzionalità LAS	1	1	0

Tabella 7.2: Matrice Funzionalità x Utente

Nell'esempio mostrato quindi:

- Utente A: può visualizzare il contenuto delle viste relative al modulo di Amministratore, al modulo della Biobanca e al modulo del LAS.
- Utente B: può accedere alle viste relative alla Biobanca e relative al LAS.

- Utente C: questo utente può solamente accedere alle viste relative al modulo della Biobanca.

Questo è da considerarsi solamente un esempio, nel progetto sono state sviluppate lato client delle funzionalità molto più specifiche, quando necessarie, legate ad un determinato modulo o a volte a un determinato componente.

7.2 Gerarchia del modello

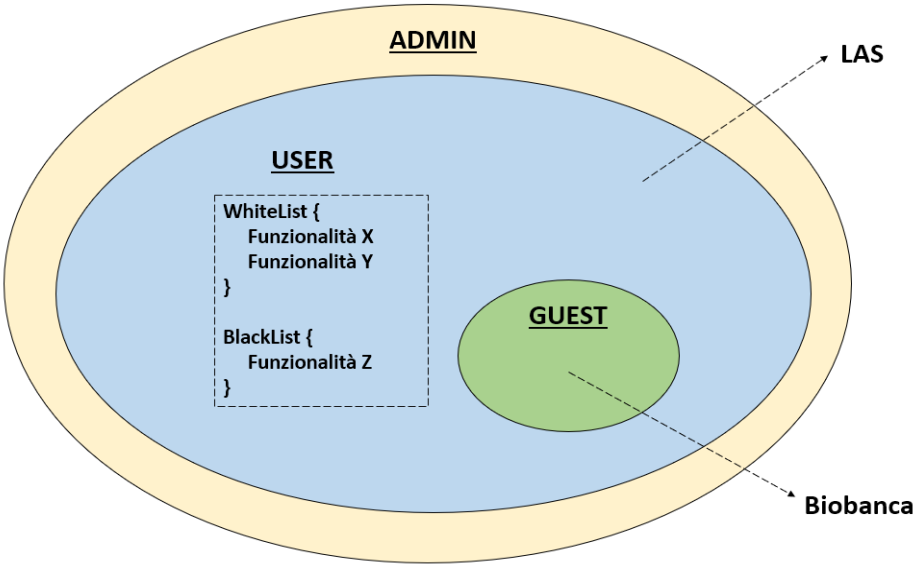

Figura 7.2: Suddivisione principale degli utenti

In via puramente teorica è possibile classificare tre principali ruoli che sono identificabili in base ai permessi associati ai dati e alle funzionalità che sono state riconosciute all'utente (Figura 7.2) .
Sostanzialmente potrebbero essere descrivibili come tre possibili funzionalità a livello macro:

- ADMIN: è il ruolo di amministratore di sistema, ha accesso alle viste che permettono la gestione degli utenti sia per quanto riguarda il LAS sia per quanto riguarda la Biobanca. All'amministratore viene concesso di accedere al contenuto intero dell'applicativo, per questo può godere di ogni funzionalità sviluppata, in quanto il suo ruolo è di monitorare anche il funzionamento dell'applicativo in caso di segnalazioni da parte degli utenti.

- USER: è un utente iscritto al LAS, può avere accesso quindi anche al contenuto della Biobanca. Possiede quindi diverse funzionalità che permettono di visualizzare o meno determinate viste del LAS. Questo utente racchiude una complessità non indifferente derivante dalla struttura del modello del LAS, costituito da Working Group, Progetti, PI, Vice PI e utenti semplici. In questo contesto quindi vivono le principali funzionalità.

- GUEST: è il ruolo di utente ospite per l'utilizzo esclusivo della Biobanca, può consultare le viste relative alla visualizzazione dei dati della Biobanca ma non può accedere alla sezione del LAS.

7.3 Creazione dei Permessi e delle Funzionalità

Un aspetto importante, successivo alla definizione dei permessi e delle funzionalità, è la loro implementazione nel codice sia a livello backend sia a livello frontend.

Il Servizio REST deve conoscere la struttura modellata e, sulla base delle relazioni che intercorrono tra le entità, concedere accesso in lettura o scrittura per un determinato dato. Questo obbliga a mantenere nel database le etichette in relazioni con le bio-entità e tutte le funzionalità associate. Durante la fase di modellazione si è cercato di mantenere minimo l'utilizzo di memoria che ne deriva da questa complessa struttura, in quanto il sistema deve garantire alte prestazioni nonostante un numero elevato di utenti, Working Group, Progetti e dati-biologici. In particolare utilizzando MongoDB e il suo Aggregation Framework è possibile utilizzare, per esempio, "$redact aggregation" il quale permette di limitare il contenuto dei documenti in base alle informazioni memorizzate nei documenti stessi. Sostanzialmente, ogni documento porta con se un insieme di etichette chiamato "Document Rights", mentre ogni utente possiederà un suo insieme di etichette chiamato "User Rights". Dall'intersezione di questi due insiemi è possibile determinare se l'utente può accedere al documento o meno. In caso di documenti innestati è possibile, facendo riferimento al risultato dell'intersezione, determinare quali sotto-documenti mostrare o meno.

Il frontend invece deve essere in grado di effettuare un rendering ottimale del contenuto, evitando di visualizzare viste o componenti quando un utente di fatto non le può utilizzare.
La complessità non è banale in quanto questo approccio rimane fattibile per quanto riguarda le tre macro funzionalità teoriche ADMIN, GUEST e USER, in quanto sono definibili in maniera statica sia per il server sia per il client. Se si vuole sviluppare un concetto di funzionalità legata al componente per impedire ad un utente, per esempio, di accedere ad una determinata "vista X" o a un "componente Y" della vista stessa, bisognerebbe trovare un meccanismo per sincronizzare le definizioni di funzionalità di accesso alla vista per il rendering con il server.

Nella pratica un gruppo di sviluppatori frontend dovrebbe aspettare che tutte le funzionalità sulla base dei casi d'uso vengano definite dagli sviluppatori backend o da chi di dovere. Quindi sulla base di un dizionario (fornito come specifica di progetto) il team creerebbe gli appositi guardiani che permettono il rendering dell'applicativo. Questo approccio, oltre a portare ad una dilatazione dei tempi in fase di sviluppo, non permette una buona manutenibilità del codice. Si immagini cosa potrebbe succedere se dovesse essere definito o rimosso un nuovo permesso o una funzionalità: il rischio che ne conseguirebbe sarebbe di avere un approccio "Hard Coding".

La soluzione adottata per mitigare questo problema è stata caratterizzata dai seguenti step:

1. Definizione del modello dei permessi associati al dato del LAS descritto in precedenza: viene utilizzato per controllare se una richiesta effettuata da parte di un utente su una API esposta dal servizio può essere soddisfatta o meno (rappresenta la gestione classica dei permessi in un Servizio REST).

2. Definizione delle funzionalità per il rendering delle viste e del loro contenuto da parte del team di sviluppo frontend: viene definita nella documentazione una linea guida per l'assegnazione del nome alla funzionalità in base al componente o modulo che si sta sviluppando in Angular (i guardiani creati per gestire il rendering delle viste utilizzano questi nomi per identificarle).

3. Integrazione delle nuove funzionalità per il rendering nel Servizio REST: viene lanciato uno script che è in grado di esaminare il codice sorgente del client e creare un piccolo dizionario delle funzionalità, le quali verranno opportunamente integrate nel servizio.

Come mostrato in Figura 7.3, il team di sviluppo può, in fase di produzione, creare nuovi moduli e componenti ai quali vengono associate nuove funzionalità. Su una determinata API è esposto un servizio che avvia uno script di analisi del codice, il quale ha il compito di determinare le funzionalità definite e aggiornare opportunamente il dizionario delle funzionalità (utilizzato per mostrare agli utenti cosa possono fare ed eventualmente richiederle all'amministratore).

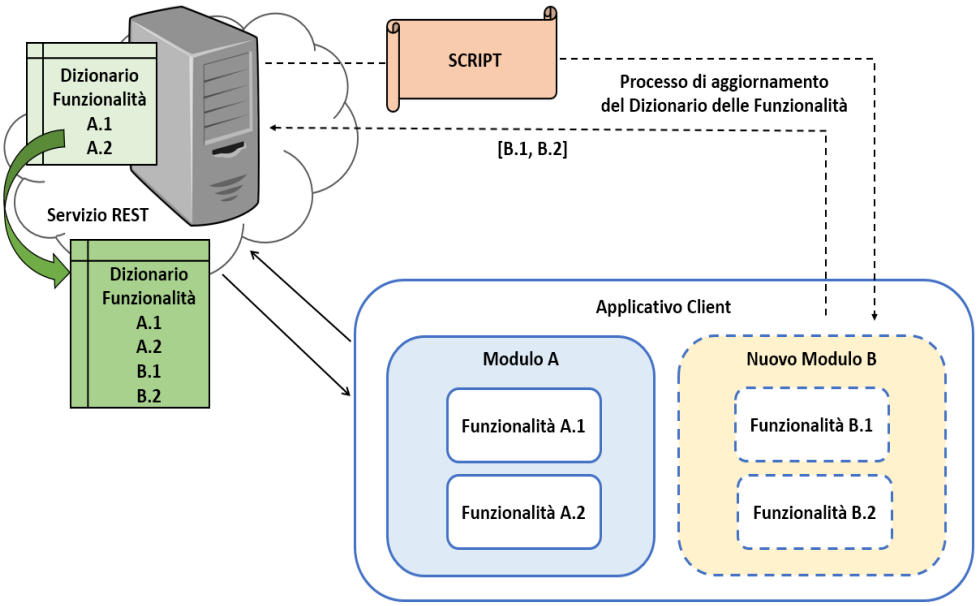

Figura 7.3: Aggiornamento del dizionario delle funzionalità

Capitolo 8

Altri moduli e servizi

In questo capitolo vengono illustrati alcuni moduli e servizi ideati per la prima versione del prototipo del progetto. Alcuni specifici del LAS 2.0 altri a carattere più generale e quindi adottati in entrambi gli applicativi. Ognuno di essi non rappresenta una soluzione definitiva ma ben si una entità in grado di crescere in autonomia e armonia con lo sviluppo dell'intero progetto. Potranno quindi evolversi arricchendosi di nuove funzionalità con lo sviluppo del progetto finale.

8.1 Modulo Amministrativo

In quanto è prevista la figura classica dell'Amministratore di Sistema, è stato sviluppato un opportuno modulo per la gestione delle sue principali attività. Uno dei compiti principali è quello di gestire il processo di registrazione dei Working Group o delle adesioni di singoli operatori ai WG. Nel capitolo 6, infatti, si era parlato del meccanismo di valutazione e approvazione da parte dell'Amministratore relativamente alle richieste di registrazione, sia per gruppi di lavoro sia di adesione singola ad un gruppo all'interno dell'applicativo del LAS.
Attraverso la sezione "WG's Management" (o Gestione dei WG) è possibile ottenere quindi una lista delle registrazioni in stato pendente e valutare l'organico del gruppo e i suoi rappresentanti, oppure, identificare singole richieste di adesione. L'Amministratore davanti ad una generica richiesta può accettarla, rifiutarla o lasciarla pendente in attesa di una valutazione futura. A seguito di una eventuale approvazione il servizio sviluppato manda in automatico l'email di attivazione account.

Un'altra caratteristica di questo modulo è la sezione "Grant Functionalities" (o Concedi Funzionalità). Nel Capitolo 7 si era indagato il modello strutturale dei permessi e delle funzionalità che possono essere erogati sulla base di moduli, componenti e dati. L'Amministratore attraverso questa interfaccia può accedere a tutte le richieste relative a determinate macro-funzionalità da parte degli utenti e valutare se garantire loro o meno determinati privilegi.

Un esempio di un caso d'uso potrebbe essere la definizione di una nuova tipologia di esperimenti molecolari. Le nuove interfacce associate per la visualizzazione e manipolazione dei dati relativi al nuovo esperimento non sarebbero accessibili ai

WG esistenti (e quindi ai suoi utenti). Nel caso in cui un solo gruppo di lavoro fosse autorizzato ad utilizzarlo si potrebbe garantire la funzionalità di accesso al nuovo modulo solo per quel WG e in questa maniera solo i suoi utenti potrebbero usufruire delle nuove schermate e lavorare sulle collezioni del database associate.

Questo modulo è destinato a crescere sulla base delle necessità dei requisiti. In generale ogni qualvolta viene creato un nuovo modulo operativo possono essere definite e aggiunte nuove sezioni al modulo amministrativo per garantirne una gestione centralizzata e corretta sotto il controllo dell'Amministratore di alcuni suoi aspetti.

8.2 Modulo JSON Editor

Questo modulo si propone di gestire la visualizzazione e la modifica di un generico contenuto in formato JSON basandosi sul Progetto sviluppato da Jos de Jong [39].

Come mostrato in Figura 8.1, è possibile esplorare il contenuto di un documento ed effettuare in maniera completamente guidata da un piccolo menu a tendina operazioni quali inserimento, duplicazione o rimozione di Array, Oggetti e Stringhe.

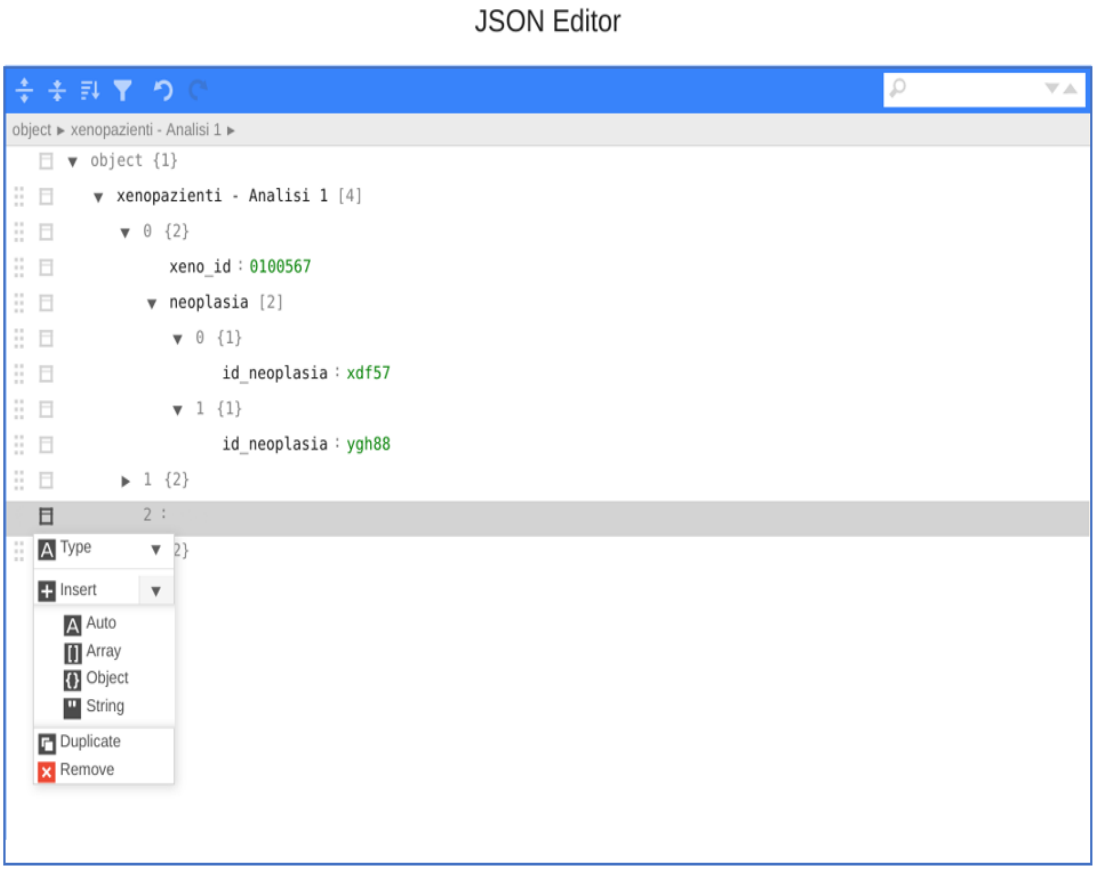

Figura 8.1: JSON Editor

Il suo utilizzo in combinazione con il servizio di cache è destinato per una eventuale gestione del contenuto delle collezioni del database principalmente in tre modi differenti:

1. accedere in sola lettura a un documento presente nel database;
2. accedere ad un documento modificandolo e salvandolo solo in locale;
3. rendere le modifiche effettuate in locale definitive tramite un'apposita API esposta dal servizio REST la quale valuta il livello di autorizzazione dell'utente relativamente ai dati e la natura dell'operazione.

In futuro si potrebbe valutare l'inserimento di un nuovo ruolo di "Database Administrator" autorizzato alla sua gestione, oppure, qualora le due figure coincidessero, verrebbe impiegato direttamente dall'Amministratore di Sistema. L'utilizzo di questo modulo garantisce di avere una gestione integrata del database nell'applicativo e permette all'Amministratore di astrarsi dalle conoscenze specifiche necessarie per manipolare il contenuto in MongoDB.

8.3 Modulo Profilo Utente

Ogni utente dispone di un modulo per la gestione del proprio profilo. Attualmente al suo interno sono state previste solamente tre sezioni:

- Account Info: permette di visualizzare le informazioni relative all'account dell'utente quali Nome, Cognome, Email, Username e Data di Iscrizione.
- Change Password: consente all'utente autenticato di accedere ad una procedura di cambio password. Per procedere bisogna inserire la vecchia password, la nuova password e la sua copia per conferma. Risulta possibile auto-generare una password (una barra cromatica indica la forza del segreto).
- Functionalities Management: permette di monitorare le proprie funzionalità e richiederne di nuove.

8.4 Modulo di Messaggistica

Per garantire la comunicazione tra gli utenti che utilizzano la piattaforma è stato previsto un modulo di supporto alla messaggistica. Attualmente l'interfaccia risulta minimale e permette l'invio e la ricezione di messaggi su uno schema identico a quello relativo alle email:

- Mittente
- Destinatario
- Eventuali Cc o Ccn (utenti che ricevono l'email per conoscenza con visibilità pubblica o privata)
- Messaggio

Un utente registrato presso la piattaforma può contattare altri utenti purché siano anch'essi registrati. Lato server è implementato un Broker in grado di assumere il ruolo di intermediario ricevendo i messaggi da parte dei client e inoltrandoli agli utenti destinatari.
Questo sistema può ancora essere arricchito: per esempio una nuova funzionalità sviluppabile in futuro potrebbe essere la possibilità di allegare file di diverso tipo al messaggio.

8.5 Paginatore

Uno dei limiti del software impiegato nel LAS 1.0 era la completa assenza di una paginazione dati. Questo comportava un trasferimento dati poco efficiente. Immaginiamo che un operatore debba accedere ad un modulo e quindi ad una tabella mostrante alcuni dati. Ora, se i dati fossero pochi l'interfaccia utente sarebbe in grado di caricare i dati senza far attendere l'utente, ma nel caso in cui i dati fossero molti di più, l'interfaccia si bloccherebbe, il cosiddetto "UI freezing" per un certo numero di millisecondi sulla base dello stato della rete e sulla base della quantità finale di dati in download. Questo forte limite incide sull'esperienza utente e sulle prestazioni del sistema in maniera drastica.

Sulla base di queste considerazioni è stato progettato un meccanismo di paginazione dei dati. Di seguito sono mostrati due esempi dei principali modelli utilizzati. Il primo è progettato per essere semplice e associato allo scambio di dati che non necessitano di essere filtrati.

```
{
  "start" : "20",
  "length" : "10",
  "count" : "43",
  "data" : [
                  {genericDocument}, ...
   ]
}
```

Come si evince dal modello il campo "start" rappresenta l'offset del numero di elementi della pagina ottenuta dal server, quindi, la prima pagina avrà sempre offset uguale a zero, ogni offset varierà in base alla lunghezza della pagina. Il campo "length" indica il numero di elementi per pagina. Per garantire una maggiore flessibilità è quindi possibile personalizzare questo campo per ottenere pagine caratterizzate da un maggior o un minor contenuto. Il campo "count" fornisce l'informazione al client della quantità totale di elementi che il server possiede e ci può fornire (in base anche ai permessi dell'utente) per quel tipo di paginazione. Infine, "data" contiene un array di documenti, risulta quindi un contenitore generico. Questo è particolarmente utile perché sulla base di un modello di paginazione unico è possibile trasferire documenti contenenti dati di natura diversa. I parametri variabili in questa tipologia di paginatore sono il campo "length", il quale consente al client di richiedere pagine di dimensioni diverse a seconda dell'esigenza e, il campo "start" il

quale consente di iniziare la pagina in un punto personalizzato eventualmente. Ricapitolando, in riferimento all'esempio mostrato sopra, se l'applicativo client ricevesse una risposta identica potrebbe dedurre che:

- In totale ci sono 43 elementi;
- Ogni pagina è costituita da 10 elementi, tranne l'ultima che sarà costituita da solo 3 elementi (in quanto 43 Mod 10 = 3);
- In totale quindi esistono 5 pagine;
- La pagina ricevuta è la terza;

Il secondo modello di paginatore, invece, rappresenta una evoluzione del primo; è stato ideato pensando ai principali casi d'uso all'interno del LAS. I parametri variabili associati a questo paginatore, oltre a "start" e "length" come per il primo, sono:

- "startFilter": rappresenta un filtro iniziale (in quanto ogni utente utilizza sempre un sotto insieme dei dati iniziali sulla base dell'interfaccia che sta utilizzando);
- "filter": consiste in un filtro ulteriore, il quale può essere personalizzato dall'utente per ottenere una vista specifica dei dati.

Quindi la struttura della risposta di questa tipologia di paginatore è costituita da due macro-oggetti "recordsTotal" e "recordsFiltered" i quali contengono i dati con relativo conteggio filtrati rispettivamente dai parametri "startFilter" e "filter". Nell'esempio che segue, possiamo evincere che si tratti della prima pagina di una collezione suddivisa in 10 pagine alla volta, il cui filtro iniziale ha restituito 5 dati mentre quello eseguito dall'utente ne ha restituiti 2.

```
{
  "start": 0,
  "length": 10,
  "recordsTotal":{
                  "data" : [ {genericDocument}, ...],
                  "count": 5
  },
  "recordsFiltered":{
                  "data" : [ {genericDocument}, ...],
                  "count": 2
  }

}
```

Bisogna sottolineare che i filtri sono implementati lato backend grazie all'Aggregation Framework offerto da MongoDB, il quale risulta un potente strumento in grado di eseguire analisi statistiche in tempo reale e generare report.

8.6 Servizio di cache

Dal punto di vista dell'applicativo client, ogni modulo e componente sviluppato in Angular può beneficiare di un servizio comune che espone un meccanismo di cache dei dati.

Il servizio si basa sul binomio Etichetta - Dato offerto dall'IndexedDB e sulla base di questa struttura offre alcune operazioni base:

- write: permette di inserire un dato associato ad una particolare etichetta;
- read: permette di leggere il dato associato a una specifica etichetta;
- clear: permette di eliminare dall'IndexedDB etichetta e dato come fossero un'unica entità.

Essendo disponibile per ogni nuovo componente o modulo che viene progettato nel sistema, è possibile impiegarlo per le interfacce che richiedono l'inserimento o la modifica di dati.

Immaginiamo che un operatore debba accedere ad una schermata e ottenga alcuni dati, di cui una parte devono essere modificati e rimandati al server. L'interfaccia può essere progettata, grazie a questo meccanismo, per reperire inizialmente dal servizio REST i dati e sulla base di ogni modifica effettuata dall'utente per salvare i risultati parziali in locale. Questo comporta il vantaggio, in caso di interruzioni improvvise di connessione, di poter lavorare offline con eventuali dati ottenuti in precedenza al fine di modificarli e salvarli momentaneamente nell'IndexedDB.

Generalmente, nel LAS, gli operatori spendono molto tempo a inserire i dati e se qualcosa andasse storto (crash del sistema o interruzione della connessione di rete) nel momento in cui vengono inviati i dati al server si rischierebbe di aver reso inutile e irrecuperabile il lavoro svolto per quelle interfacce progettate senza il meccanismo di recupero di sessione.

In questa maniera i componenti operativi possono essere progettati con un meccanismo personalizzato di recovery che sulla base di un qualsiasi problema possa fornire il ricaricamento e la rigenerazione delle schermate popolate dai dati inseriti in precedenza in maniera semplice e immediata per l'utente.

Capitolo 9

Aspetti legali

L'attività fondamentale di una biobanca non è quella di svolgere direttamente attività di ricerca, ma bensì di rendere disponibile ai ricercatori i bio-materiali indispensabili alla ricerca stessa.
L'istituto, il quale ha in carico la gestione completa della biobanca dati, si trova a ricoprire il ruolo di intermediario tra i donatori ed i ricercatori: assume il compito di Broker Onesto.
Un broker onesto è un'entità il cui compito è quello di conservare un'insieme di informazioni private relative a determinati soggetti, le quali vengono successivamente ripartite ad altre entità secondo opportune regole (ogni entità accede solo ad un sotto-insieme dei dati).
Questo modello è ampiamente diffuso nell'ambito della ricerca clinica e della raccolta di campioni biologici. I donatori di campioni consentono ai ricercatori di effettuare ricerche sui loro bio-materiali, ma in genere desiderano che il loro campione venga de-identificato avendo separato le informazioni sanitarie da esso (in questa maniera si rende anonimo il dato). Il broker onesto mantiene sia il campione che le informazioni sulla salute protette associate, ma consente ai ricercatori di avere accesso solamente al campione senza le informazioni sanitarie protette.

In generale, la disponibilità di biomateriali organizzati in collezioni per finalità di ricerca permette di effettuare un primo studio, innestando, principalmente, le aliquote derivanti dai campioni estratti da un paziente umano a un topo di laboratorio. Sulla base delle ricerche e delle sperimentazioni di terapie effettuate sugli animali vengono pubblicati risultati. Successivamente, attraverso iter complessi che richiedono l'accettazione della comunità scientifica, vengono avviate opportune sperimentazioni cliniche sull'uomo. Sarà oggetto di questo Capitolo soffermarsi su alcuni aspetti legali relativi alla raccolta, conservazione e utilizzo dei dati personali, clinici e biologici dei pazienti.

9.1 Sperimentazione clinica

Una sperimentazione clinica è un tipo di studio condotto per raccogliere dati sulla sicurezza e sull'efficacia di nuove terapie o di nuovi dispositivi medici. Le sperimentazioni possono essere condotte solo dopo che siano state raccolte tutte le informazioni

necessarie sulle caratteristiche della terapia/procedura e sulla sua sicurezza.

Presso l'Istituto di Candiolo [40] è attivo un Comitato Etico di natura indipendente. Il suo compito principale è quello di approvare e monitorare le sperimentazioni cliniche dei farmaci, oltre ad espletare funzioni consultive in merito alle questioni etiche inerenti le attività scientifiche ed assistenziali. Per quanto riguarda l'ambito delle sperimentazioni di farmaci e dei dispositivi medici, assume una rilevanza particolare il ruolo di garanzia del Comitato Etico per la salvaguardia dei principi di riservatezza, informazione consapevole e sicurezza clinica dei pazienti. Il Comitato Etico assume il ruolo di garante per la tutela del benessere, della sicurezza e dei diritti dei soggetti che partecipano ad una sperimentazione clinica.

Ogniqualvolta è necessario avviare una nuova tipologia di sperimentazione, questa viene valutata nel suo complesso, in particolare vigilando sulla sussistenza dei requisiti in materia di consenso informato, informativa privacy, copertura assicurativa e contenuto innovativo della sperimentazione.
Per procedere ad una sperimentazione clinica è necessaria l'approvazione del comitato etico. Vengono reclutati esclusivamente i pazienti che sottoscrivono lo specifico consenso informato.

9.2 Raccolta dati

In generale ogni paziente assistito all'interno del centro attraversa un diverso percorso clinico. Sulla base del suo percorso (esami diagnostici, terapie e interventi chirurgici) vengono raccolte sia informazioni a carattere personale sia a carattere clinico. Ogni paziente può autorizzare l'utilizzo di queste per finalità di ricerca tramite una clausola del consenso informato.

Il "consenso informato" è la manifestazione di volontà mediante la quale il paziente esprime liberamente il suo consenso o dissenso in merito a un trattamento terapeutico da seguire attraverso una informazione esaustiva sulle sue condizioni di salute e, soprattutto, sulle conseguenze e i rischi connessi alla terapia stessa . Il termine "consenso informato" è nato a seguito del processo di Norimberga, quando l'omonimo codice evidenziò il principio dell'inviolabilità della persona umana: la partecipazione di qualunque individuo ad una ricerca scientifica non sarebbe più avvenuta senza il suo volontario consenso [41].

Il consenso informato è considerato valido se detiene le seguenti caratteristiche:

1. Personale: deve essere espresso dal soggetto sul quale si basa il consenso stesso, tranne nei casi di incapacità giuridica (per quanto riguarda minori, persone soggette a condanne o non capaci di intendere e di volere opportunamente assistite e rappresentate);

2. Libero: il consenso informato deve essere firmato in assenza di manipolazioni di alcun genere da parte di altri soggetti (pressioni psicologiche o minacce);

3. Esplicito: manifestato in maniera chiara e inequivocabile;

4. Consapevole: il soggetto firmatario del consenso informato deve disporre di tutte le informazioni necessarie per effettuare la propria decisione;
5. Specifico: è modellato sulla base della natura delle azioni svolte;
6. Revocabile: può essere revocato in ogni momento.

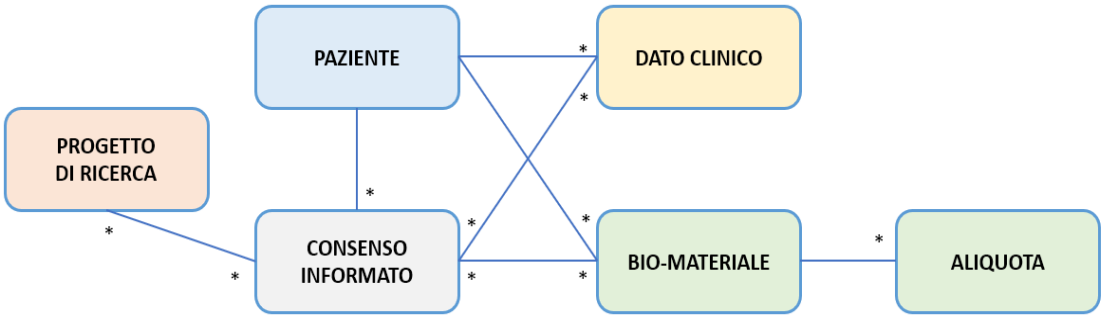

Figura 9.1: Modellazione Consenso Informato

Per la disciplina delle biobanche, in riferimento a [42], il consenso informato firmato dal donatore è relativo sia ai dati clinici utilizzabili nella biobanca sia ai campioni di bio-materiali estratti a seguito di esami o interventi chirurgici, nel loro completo o anche solamente per un sottoinsieme di essi (Figura 9.1).
In generale un paziente può firmare più di un consenso informato, tali consensi possono essere relativi allo stesso progetto o a progetti di ricerca differenti.

A livello internazionale sono previsti alcuni modelli di riferimento principali:

- Consenso ampio: permette l'utilizzo di campioni e di dati associati in ricerche presenti e future di ogni tipologia;
- Consenso parzialmente ristretto: consente l'utilizzo di campioni e di dati associati per una ricerca immediata specifica e analisi future associate ai campioni utilizzati;
- Consenso multi-opzione: consiste di diverse opzioni, ognuna delle quali deve essere riportata al donatore in maniera trasparente;
- Consenso informato specifico: permette l'utilizzo di campioni e di dati associati limitatamente per una o più ricerche specifiche.

[42] In sede internazionale, particolarmente europea, gli elementi essenziali dell'informazione per il donatore da registrare con modalità analitiche nella documentazione dell'avvenuto consenso o dissenso sono:

- la partecipazione volontaria;

- l'eventuale trasferimento dei campioni ad altra banca, o a gruppi di ricerca diversi dal proponente;
- la possibilità o l'esclusione di un ritorno d'informazione al donatore sui risultati della ricerca, (esclusione che si può realizzare quando l'indagine sul materiale genetico non ha un significato "clinico");
- le indicazioni sulle possibili conseguenze per il donatore o i membri della sua famiglia dei risultati delle analisi genetiche;
- la possibilità di rendere anonimi i campioni o di identificarli con un codice;
- le misure adottate per la tutela dei dati personali;
- la possibilità per il donatore di revocare, in ogni momento, il proprio consenso;
- il destino dei campioni in caso di revoca o di chiusura della biobanca;
- le eventuali prospettive commerciali della ricerca (compreso il deposito di eventuali brevetti) e degli eventuali rimborsi per le spese sostenute, nonché della partecipazione del donatore agli eventuali benefici diagnosti e terapeutici derivanti dalla ricerca.

9.3 Privacy

Per quanto riguarda i dati raccolti, relativi sia ai pazienti assistiti dall'istituto sia agli utenti registrati ad una delle applicazioni sviluppate, occorre analizzare il il regolamento regolamento (UE) n. 2016/679 in merito alla protezione dei dati e alla privacy (in inglese General Data Protection Regulation o GDPR).
Il GDPR è entrato in vigore il 25 maggio 2018 e, attraverso di esso, la Commissione europea vuole migliorare e generalizzare il processo di protezione dei dati personali dei cittadini e dei residenti dell'Unione Europea, sia all'interno che all'esterno dei confini UE.

[43] I nuovi principi introdotti sono:

1. Privacy by design (in riferimento all'art.25): rappresenta l'onere, per chi costruisce un database di informazioni personali, di organizzarlo e strutturarlo sulla base degli obblighi imposti dalla nuova normativa. In sostanza, i problemi relativi alla riservatezza dei dati devono essere prevenuti attraverso una buona progettazione della base di dati.
2. Privacy by default (in riferimento all'art.25): la quantità di dati raccolti ed il tempo di loro conservazione non deve eccedere il minimo necessario all'espletamento delle finalità perseguite dal trattamento.
3. Principio di responsabilità (in riferimento all'art.24): il titolare deve adottare a sua discrezione politiche e misure adeguate per garantire e dimostrare che il trattamento dei dati personali è conforme a quanto previsto dalla nuova normativa.

4. Notifica di violazione (in riferimento all'art.33): in caso di violazione dei dati personali, il titolare del trattamento deve notificare tale violazione all'autorità di controllo competente entro 72 ore dal momento in cui ne è venuto a conoscenza, a meno che sia improbabile che la violazione dei dati personali presenti un rischio per i diritti e le libertà delle persone fisiche. Se la comunicazione avviene in un tempo successivo a quanto stabilito occorre allegare una motivazione del ritardo.

Le modalità con cui vengono raccolti e trattati i dati devono basarsi sul consenso del soggetto interessato. In particolare tale consenso deve avvenire in un momento precedente all'acquisizione dei dati, deve essere esplicito e informato (devono essere specificati come verranno raccolti e utilizzati i dati, per quanto saranno disponibili, le finalità del loro trattamento ed eventuali diritti dell'utente). Inoltre, il titolare del trattamento deve essere in ogni momento in grado di dimostrare l'avvenuto consenso da parte dell'interessato.

[44] Per l'intestatario, nel caso di un sito web, sono previsti tre diritti circa i dati di sua pertinenza:

- Diritto all'accesso: l'interessato deve poter accedere ai propri dati, al fine di poterne verificare la correttezza ed, eventualmente, modificarli.
- Diritto all'oblio: l'interessato può richiedere la cancellazione dei propri dati in modo semplice o automatizzato tramite apposite funzioni on-line.
- Diritto di portabilità: l'interessato è in grado di ottenere una copia di tutti i propri dati al fine di poterli portare presso un altro gestore di servizio.

Per quanto riguarda i CMS come Wordpress, utilizzato in questo progetto per la creazione di un Blog, risulta opportuno effettuare aggiornamenti periodici della versione utilizzata e degli eventuali plugin installati.

L'obiettivo principale del GDPR è quindi quello di unificare la legislazione in merito alla protezione dei dati personali all'interno dell'Unione Europea. Da un lato i cittadini possiedono un maggior controllo della tutela dei loro dati, dall'altro la loro circolazione viene massimizzata attraverso il processo di digitalizzazione.
Dal punto di vista della ricerca i dati biologici rappresentano la base per la creazione di nuove terapie e l'aggiornamento della conoscenza medica.
La domanda che si pongono i due scienziati italiani Marelli e Testa in un articolo pubblicato sulla rivista Science è la seguente:

> "Dato che ogni progetto di ricerca genera dati personali, in formato digitale, riutilizzabili per un altro studio con finalità differenti, e dato che per le informazioni personali viene sempre richiesto il consenso informato, come bisogna agire nel momento in cui si permetta la circolazione dei dati in contesti differenti? O viene richiesto ogni volta il consenso alla persona oppure viene richiesta una tipologia di consenso allargato per l'utilizzo dei dati in ricerca da parte della stessa istituzione.

> Ma ora si pone un altro problema: cosa accadrebbe qualora il dato passasse di mano, ed entrasse in possesso di un'altra istituzione, con magari un profilo istituzionale completamente differente da quella iniziale?" [45]

[46] Come osservato dai due studiosi, la questione dell'uso e del riutilizzo dei dati personali per fini di ricerca scientifica viene affrontata dal nuovo regolamento europeo in modo ambivalente.
Da un lato, il GDPR introduce tutta una serie di facilitazioni e deroghe per il trattamento di dati personali per scopi di ricerca scientifica, fornendo un'interpretazione molto ampia di cosa costituisca "ricerca scientifica", fino a comprendere la ricerca svolta con fondi privati e per finalità non necessariamente volte al perseguimento del pubblico interesse. Dall'altro lato, il GDPR pone le basi per la promozione degli interessi e delle istanze dei pazienti e dei partecipanti alla ricerca, richiedendo, prima di poter riutilizzare dati personali, di condurre un'analisi del rapporto fra l'ente che cede il dato e quello che lo acquisisce, verificando che gli obiettivi e i valori delle due istituzioni siano compatibili.

In conclusione, il nuovo regolamento GDPR viene considerato una soluzione flessibile per la regolamentazione e lo sviluppo della ricerca scientifica nel rispetto di una corretta gestione dei dati raccolti. Si focalizza sull'importanza del ruolo assunto dagli organismi regolatori della ricerca biomedica, come per esempio comitati etici e istituzioni scientifiche impegnate nella creazione di specifici codici di condotta, oltre a promuovere una grande responsabilizzazione collettiva circa la tutela e la circolazione dei dati personali [46].
Tuttavia, in questo contesto, rappresenta anche uno strumento che può essere soggetto a superflue interpretazioni e a una cattiva gestione. Sulla base di queste considerazioni è da considerarsi un punto fondamentale durante la modellazione e la creazione del progetto.

L'adeguamento al GDPR conivolge aspetti tecnici spesso complessi e richiede quindi interventi da parte di diverse figure professionali come consulenti legali, sistemisti e sviluppatori. Questo aspetto richiede un'attenta analisi costante durante il processo di creazione della nuova piattaforma.

Capitolo 10

Conclusione

10.1 Obiettivi raggiunti

L'architettura base del progetto è stata ben definita e sono state scelte le tecnologie costitutive principali. La modernizzazione, l'efficienza e la scalabilità sono state le pietre miliari durante l'analisi e la scelta degli strumenti da impiegare.
Il modello dati si basa sull'utilizzo di un unico database NoSQL, questo approccio riduce la complessità di gestione e aumenta la flessibilità in quanto non occorre definire alcun schema fisso.
Il framework, basandosi sulla modularità sia dal punto di vista del servizio REST sviluppato sia per quanto riguarda l'applicativo client, è progettato per permettere una continua evoluzione nel tempo dell'intero sistema e una semplice manutenibilità da parte del team di sviluppo.
Ad esempio, nel caso in cui fosse necessario creare una nuova funzionalità occorre implementare un micro-servizio lato server che esponga determinate API e che scambi dati tramite di esse con un modulo client sviluppato ad-hoc.
Mentre, per quanto riguarda una modifica, basta agire sul codice direttamente nel modulo interessato sul backend e/o frontend, senza dover modificare il resto dell'applicativo.

Per gli applicativi del LAS e della Biobanca sono stati implementati, come mostrato nei capitoli precedenti, alcuni meccanismi base quali l'autenticazione, la gestione dei permessi, il modulo di amministrazione, il supporto alla messaggistica e alcuni servizi e componenti a carattere generale che possono trovare un largo impiego e riutilizzo nel framework.
I punti fondamentali nella gestione del processo di autenticazione e degli utenti è rappresentato dal mantenimento di uno stato lato client permesso grazie al modello di programmazione "Redux". In questo modo è possibile risalire alle informazioni principali di un profilo utente e gestire il rendering delle interfacce visualizzate.
Inoltre, è stato predisposto un servizio di cache in grado di permettere una memorizzazione locale delle modifiche effettuate sui dati dal frontend utile per un eventuale funzionamento off-line dell'applicazione o per recuperare i dati in caso di errori imprevisti.

Per quanto riguarda invece il Blog sono state create alcune pagine web per la presentazione e pubblicizzazione del progetto nel suo intero. La sua peculiarità è la semplicità: rispetto agli altri due applicativi è pensato per essere gestito, sviluppato e mantenuto da utenti che non devono necessariamente essere in possesso di capacità particolari. Il suo contenuto, nell'ottica del processo di distribuzione, è pensato per essere personalizzato direttamente da un responsabile dell'istituzione coinvolta. Quella che è stata predisposta, quindi, è un'ottima base di partenza per un sito web "vetrina" opportunamente configurabile da chi di dovere.

L'intero progetto è organizzato e gestito da una apposita tecnologia di virtualizzazione in grado di permetterne la distribuzione e la diffusione.
Il corretto funzionamento di quanto sviluppato è stato opportunamente controllato attraverso strumenti di debug e test.
Infine, è stata realizzata una completa documentazione del sistema REST e degli applicativi. Questo rappresenta un'utile pratica per permettere a sviluppatori che non conoscono a fondo il progetto e ogni sua parte di implementare agilmente nuove funzionalità in breve tempo.

10.2 Obiettivi futuri

Il progetto allo stato attuale rappresenta un prototipo. Per il prossimo futuro possono essere individuate due categorie principali di obiettivi: a livello implementativo e a livello istituzionale.
Per quanto riguarda i primi occorre sicuramente rendere il prototipo totalmente operativo e funzionante, destinandolo quindi alla distribuzione. Ora che è stata creata la base sulla quale operare è possibile lavorare in parallelo sui progetti del LAS e della Biobanca per completarli. Per quanto riguarda il primo bisognerà adattare i moduli esistenti del LAS 1.0 per il nuovo applicativo ed eventualmente implementare nuove funzionalità.

Verrà predisposto un modulo di gestione degli Xenopazienti, il quale permette di monitorare e di controllare il ciclo di vita degli xenopatienti, dal processo di acquisizione da parte dell'istituto di ricerca fino alla momento della loro morte. In particolare si occupa della gestione delle colonie di topi presenti nel sistema, della gestione degli impianti effettuati sugli xenopazienti, del monitoraggio della crescita dei tumori associati e della gestione degli espianti [47].

Il modulo di gestione dello Storage avrà il compito di rappresentare e gestire l'archiviazione di materiale biologico avvenuto in appositi contenitori del laboratorio (come freezer, piastre e tubi). La diversità delle tipologie di contenitore implica dover utilizzare uno schema flessibile per i dati (il modello NoSQL di MongoDB si presta perfettamente a risolvere la complessità di questo problema).

Fondamentale sarà la presenza di un modulo di gestione degli esperimenti genomici, il quale permetterà di gestire, portare a termine e monitorare gli esperimenti eseguiti dagli operatori. In particolare NGS (Next Generation Sequencing), biopsia dei liquidi organici e RTPCR (vedi Capitolo 2).

Per finire, per quanto riguarda sempre il LAS 2.0, verrà reingegnerizzato il precedente modulo per l'analisi multidimensionale dei dati, il quale fornisce uno strumento

per interpretare correttamente i dati biologici e molecolari e per scoprire nuove conoscenze relative ai tumori. Il suo scopo è di fondamentale importanza e prevede l'estrazione di ogni informazione ottenuta in laboratorio e memorizzata nella banca dati sfruttando diversi strumenti grafici. L'utente, in pratica, pianifica un insieme di interrogazioni (workflow) per filtrare opportunamente i dati e ottenere conoscenza [48].

A proposito della Biobanca, invece, lo scopo è di tradurre parte delle funzionalità del sotto-modulo omonimo del vecchio progetto e determinare nuovi casi d'uso e interfacce. In particolare potranno essere sviluppati due moduli differenti: uno per la gestione dei tessuti e uno per la gestione delle informazioni genetiche. I compiti di entrambi coinvolgono il processo di collezione dei materiali biologici a seguito di interventi sui pazienti e il processo di acquisizione di aliquote da laboratori esterni. Inoltre, prevedono la misurazione delle principali caratteristiche fisiche di ogni aliquota (ad esempio volume, concentrazione e qualità) e il monitoraggio del consumo dei materiali raccolti causato dalle operazioni di laboratorio o dal normale deterioramento.

Cambiando completamente discorso, per quanto riguarda gli obiettivi a carattere istituzionale emersi durante la creazione di questo progetto, in un prossimo futuro la Biobanca ideata e la sua distribuzione potrebbero aderire ad una rete su vasta scala in quanto la Commissione Europea ha finanziato numerosi progetti di collaborazione i quali coinvolgono diverse biobanche. In particolare il BBMRI.it è il Nodo Nazionale della Infrastruttura di Ricerca Europea delle Biobanche e delle Risorse BioMolecolari (BBMRI-ERIC), è nato grazie all'impegno congiunto del Ministero dell'Università e della Ricerca e del Ministero della Salute. All'infrastruttura contribuiscono istituzioni di ricerca, quali l'Istituto Superiore di Sanità, il Consiglio Nazionale della Ricerca, 23 Istituti di Ricovero e Cura a Carattere Scientifico (IRCCS), 18 Università, 40 Aziende Ospedaliere, 290 ricercatori e gruppi di ricerca dell'università e del CNR. Inoltre, un network di stakeholders, che include 8 associazioni di pazienti tra cui Uniamo, Federazione Italiana Malattie Rare e FAVO, Federazione Italiana delle Associazioni di Volontariato in Oncologia, aziende in ambito biomedico e biotecnologico e associazioni scientifiche, supporta e collabora con il nodo per definire obiettivi e fornire expertise [49].

Questa moderna infrastruttura è pensata per coordinare le attività delle biobanche a livello italiano permettendone una buona interoperabilità anche livello comunitario. Uno dei punti forza è il collegamento tra ricerca biomedica e l'attività clinica nella rete IRCCS.
Le tipologie di biobanche partecipanti a questo progetto sono principalmente di tipo clinico, di popolazione e di tessuti. Le biobanche cliniche prevedono la raccolta di campioni biologici (quali tessuti, cellule e DNA) a cui vengono associati dati epidemiologici, clinici e di ricerca. Il loro scopo è quello di determinare nuovi strumenti diagnostici, nuove terapie e nuovi farmaci. Le biobanche di popolazione hanno principalmente le stesse funzionalità delle banche cliniche ma a carattere più generale. Le biobanche di tessuti, invece, collezionano campioni con dati clinici e biologici utilizzabili per studi retrospettivi.

BBMRI intende inoltre sviluppare linee guida condivise per gli aspetti etico-legali

riguardanti la raccolta, la conservazione e l'uso dei campioni. In generale ogni biobanca partecipante viene connessa alla rete europea mediante il lavoro svolto da coordinatori nazionali, che relativamente al loro paese di origine, finanziano la rete statale e propongono eventuali interventi legislativi.

L'adesione a questo moderno network europeo delle biobanche potrebbe rappresentare un'ottima iniziativa per inserire il progetto in un contesto molto più ampio e attraverso di esso potrebbe essere possibile raggiungere biobanche emergenti facendo loro beneficiare delle tecnologie sviluppate, tramite opportuni accordi, attraverso il processo di distribuzione del software previsto per questo framework.

In conclusione, gli obiettivi futuri a carattere implementativo saranno pianificati dal team di sviluppo e raggiunti nel breve periodo per rendere il framework totalmente operativo all'interno dell'Istituto di Candiolo IRCCS, in modo da avviare il processo di sostituzione e distribuzione del software.
Per quanto riguarda gli obiettivi a carattere istituzionale, questi rappresentano solo un'ipotesi effettuata durante la stesura del presente documento. Il futuro di questo scenario potrà eventualmente essere preso in considerazione dagli organi amministrativi dell'istituto, valutandone vantaggi e svantaggi connessi.

Bibliografia

[1] A.Fiori, A.Grand, P.Alberto, E.Geda, F.G.Brundu, D.Schioppa, A.Bertotti, "Laboratory Management Information Systems: Current Requirements and Future Perspectives", IGI Global, 2015, DOI 10.4018/978-1-4666-6320-6.ch013

[2] E.Baralis, A.Bertotti, F.Bussolino, A.Fiori, A.Grand, E.Medico, T.Renzulli, "Laboratory Assistant Suite platform for biomedical data management and integration", Congresso Nazionale AICA, January, 2011, DOI 10.13140/RG.2.1.1859.3768.

[3] Bertotti A., Migliardi G., Galimi F. et al., A molecularly annotated platform of patient-derived xenografts ('xenopatients') identifies HER2 as an effective therapeutic target in cetuximab-resistant colorectal cancer, Cancer Discovery (2011).

[4] Sequenom, fondata da Hubert Köster https://www.sequenom.com/

[5] Paik H., Lemos A.L., Barukh M.C., Benatallah B., Natarajan A. (2017) Web Services – REST or Restful Services. In: Web Service Implementation and Composition Techniques. Springer, Cham DOI 10.1007/978-3-319-55542-3_3

[6] Presidenza del Consiglio dei Ministri, Comitato Nazionale per la Biosicurezza e le Biotecnologie, Linee guida per l'istituzione e l'accreditamento delle Biobanche, Rapporto del Gruppo di lavoro, 19 Aprile 2006.

[7] Apache Software Foundation, Apache, https://www.apache.org/

[8] NGINX, Nginx, Inc. https://www.nginx.com/

[9] Kubernetes, https://kubernetes.io/

[10] Docker, https://www.docker.com/

[11] Docs.microsoft.com. (2018). Guida introduttiva alla modalità Swarm. https://docs.microsoft.com/it-it/virtualization/windowscontainers/manage-containers/swarm-mode

[12] MySQL, https://www.mysql.com/

[13] MongoDB https://www.mongodb.com/

[14] Neo4J, https://neo4j.com/

[15] Neo4j Graph Database Platform. (2018). Neo4j and MongoDB https://neo4j.com/developer/mongodb/

[16] Spring, https://spring.io/

[17] Django, https://www.djangoproject.com/

[18] P.Skolski, (2016), Single-Page Application vs Multiple-Page Application-Neoteric. https://neoteric.eu/single-page-application-vs-multiple-page-application

[19] AngularJS, https://angularjs.org/

[20] Angular, https://angular.io/

[21] React, https://reactjs.org/
[22] Krajka, B. (2015). The difference between Virtual DOM and DOM. http://reactkungfu.com/2015/10/the-difference-between-virtual-dom-and-dom/
[23] Vue.js https://vuejs.org/
[24] Portainer, https://portainer.io/
[25] Redux https://redux.js.org/
[26] Wordpress https://it.wordpress.org/
[27] WiredTiger http://www.wiredtiger.com/
[28] MongoDB, MongoDB Compass GUI, https://www.mongodb.com/products/compass
[29] Swagger https://swagger.io/
[30] E. Rescorla, "HTTP Over TLS", RFC-2818, May 2000, DOI 10.17487/RFC2818
[31] Y. Sheffer, Porticor, R. Holz, Technische Universitaet Muenchen, P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC-7457, February 2015, DOI 10.17487/RFC7457
[32] T.Dierks, E.Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC-5246, August 2008, DOI 10.17487/RFC5246
[33] D. Cooper, NIST, S. Santesson, Microsoft, S. Farrell, Trinity College Dublin, S. Boeyen, Entrust, R. Housley, Vigil Security, W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" RFC-5280, May 2008, DOI 10.17487/RFC5280
[34] M. Jones, J. Bradley, N. Sakimura, "JSON Web Token (JWT)", RFC-7519, May 2015, DOI 10.17487/RFC7519
[35] The Open Web Application Security Project, "Cross-site Scripting (XSS)", https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
[36] Google, DomSanitizer, https://angular.io/api/platform-browser/DomSanitizer
[37] The Open Web Application Security Project, "Content Security Policy", https://www.owasp.org/index.php/Content_Security_Policy
[38] The Open Web Application Security Project, "Cross-Site Request Forgery (CSRF)", https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
[39] Jos de Jong, JSON Editor Online, http://jsoneditoronline.org
[40] Candiolo Cancer Institute - IRCCS, http://www.ircc.it/irccit/
[41] M.C.Piegari, Il Consenso Informato, 30 Maggio 2018, https://www.nurse24.it/studenti/risorse-studenti/il-consenso-informato.html
[42] Presidenza del Consiglio dei Ministri, Comitato Nazionale di Bioetica, Comitato Nazionale per la Biosicurezza, le Biotecnologie e le Scienze della Vita, Raccolta di campioni biologici ai fini di ricerca: Consenso Informato, 16 Febbraio 2009.
[43] Bossi, M. Mr. Webmaster. GDPR e siti web: cosa fare per essere in regola con la privacy. 12 Maggio 2018. https://www.mrwebmaster.it/leggi-fisco/gdpr-cosa-fare-sito-web_12510.html

[44] Garante per la protezione dei dati personali, https://www.garanteprivacy.it/

[45] L.Marelli, G.Testa, Scrutinizing the EU General Data Protection Regulation, Science (04 May 2018), Vol. 360, Issue 6388, pp. 496-498, DOI 10.1126/science.aar5419

[46] Università degli studi di Milano, Istituto Europeo di Oncologia, (2018), http://www.unimi.it/lastatalenews/sites/default/files/attachments/CSScience_GDPR.pdf

[47] El Akadi A., Amine A., El Ouardighi A. et al., A two-stage gene selection scheme utilizing MRMR filter and GA wrapper, Knowledge and Information Systems. (2010)

[48] Golub TR, Slonim DK, Tamayo P. et al., Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science, 286, 5439, 531 (1999)

[49] Biobanking and BioMolecular Resources Research Infrastructure of Italy, https://www.bbmri.it/

Appendice A

Definizioni

In questa appendice sono presenti le definizioni di alcuni aspetti e terminologie utilizzati nel presente documento per permetterne una più profonda comprensione.

A.1 Acidi Nucleici

Gli acidi nucleici sono composti organici formati da molecole di grandi dimensioni e complessità. Il loro nome deriva dal fatto che sono presenti nel nucleo cellulare di tutti gli organismi viventi. Il loro compito principale è quello di direzione e controllo della sintesi delle proteine all'interno di un organismo vivente. Inoltre, assumono un ruolo di fondamentale importanza biologica in quanto sono depositari dell'informazione genetica e della sua trasmissione di generazione in generazione. Esistono due tipi di acidi nucleici: l'acido desossiribonucleico o DNA e l'acido ribonucleico o RNA.

A.2 Aliquota

L'aliquota è una frazione rappresentativa (o porzione) del campione di laboratorio. Questa può essere:

- impiantata nei topi: questa operazione ha come risultato la generazione di uno xenopaziente.
- derivata: vengono generate nuove aliquote a partire da quella base.
- suddivise: l'aliquota viene suddivisa in sotto-aliquote identiche.

A.3 Biobanca

La piattaforma della Biobanca eroga un servizio, senza scopo di lucro diretto, il cui scopo principale è quello di raccogliere e conservare campioni di biomateriale

umano. I suddetti campioni vengono associati a dati clinici e a informazioni derivanti da analisi approfondite nei laboratori, con la finalità di fornire un supporto per la ricerca riconosciuta dalla comunità scientifica. La normativa associata allo sviluppo e mantenimento di una biobanca richiede che i campioni siano:

- prelevati e preservati secondo procedure idonee a garantire un'ottima conservazione dei componenti strutturali (istologici e biochimici).
- raccolti, conservati ed utilizzati seguendo opportuni criteri di bioetica e biodiritto;

A.4 Bio-entità

Viene definita Bio-entità un qualsiasi insieme di informazioni associate a un determinato Biomateriale estratto da un soggetto umano o da una cavia da laboratorio. La Bio-entità è quindi la rappresentazione digitale, sotto forma di dato, del biomateriale. Viene collezionata e resa persistente nella base dati comune per LAS e Biobanca.

A.5 Biomateriale umano

Viene definito biomateriale umano un campione di liquidi biologici e/o tessuti prelevati da soggetti umani, sia che essi siano sani sia che essi siano affetti da malattia. Da tali campioni è possibile estrarre le componenti biochimiche fondamentali: l'RNA, il DNA e le proteine. Il biomateriale permette di accedere al contenuto nel genoma umano, con la possibilità di estrarre un "profilo genetico" della singola persona. In questo contesto è essenziale che il materiale sia raccolto e conservato rispettando le normative sulla sicurezza e sulla tutela dei dati personali.

A.6 Blog

Rappresenta un sito utilizzato dall'amministratore di sistema per fornire informazioni di base relative al lavoro di ricerca svolto dalle istituzioni coinvolte. Descrive l'operato del LAS e indica come accedere eventualmente alle collezioni di dati presenti nella Biobanca. Può essere utilizzato per rilasciare periodicamente dei post con la finalità di aggiornare la comunità di ricerca sulle attività svolte e su quelle future.

A.7 Collezione

Costituisce un aggregato di diversi campioni di biomateriale che condividono caratteristiche comuni. Ad esempio, una raccolta può riguardare tutti i tessuti raccolti da un paziente dopo un intervento chirurgico.

A.8 Consenso Informato

Rappresenta una forma di autorizzazione del paziente la quale permette l'utilizzo dei campioni di biomateriale e dei dati clinici per lo studio e l'analisi con finalità di ricerca scientifica e medica.

A.9 Espianto

Viene definito espianto un campione di liquidi biologici e/o tessuti prelevati da cavie da laboratorio, relativamente sia da soggetti sani sia da soggetti affetti da malattia. Da tali campioni è possibile estrarre le componenti biochimiche fondamentali: l'RNA, il DNA e le proteine.

A.10 Fenotipo

Con questo termine si intende l'insieme delle caratteristiche morfologiche e funzionali manifestate da un organismo vivente.

A.11 Genealogy ID

Il genealogy ID è un codice univoco che viene rilasciato solo per alcune bio-entità. Si tratta di un'alias dell'entità al fine di mantenere una retro-compatibilità e fornire le informazioni di base e d'interesse a un utente durante la manipolazione di queste. Le regole di generazione dipendono dalle regole di definizione delle bio-entità che hanno diritto di essere etichettate con un apposito genealogy ID.

A.12 Genoma

Rappresenta l'intero insieme dei geni di una cellula o di un organismo.

A.13 Impianto

Viene definito impianto il processo relativo all'immissione di una aliquota all'interno del sistema biologico di una cavia da laboratorio ai fini di studiarne gli effetti derivanti.

A.14 Laboratory Assistant Suite

La piattaforma LAS (Laboratory Assistant Suite) assiste i ricercatori in diverse attività di laboratorio. La sua architettura modulare consente di gestire diversi tipi di dati grezzi (ad es. Biologici, molecolari) e di tracciare dati sperimentali. Ogni modulo LAS è progettato per gestire attività o tipi di dati specifici, ma è inserito in un framework più ampio e uniforme, consentendo così un'integrazione senza sforzo con gli altri elementi del sistema. I modelli di dato e le procedure integrati nella piattaforma cercano di conformarsi alle migliori pratiche e standard ampiamente adottati dalla comunità di ricerca in generale. Le interfacce utente sono progettate per essere pratiche in ambienti ostili, in cui i ricercatori dovrebbero minimizzare le loro interazioni con il sistema durante le procedure di immissione dei dati (ad es. In condizioni sterili). Inoltre, la piattaforma supporta l'integrazione di diverse risorse e aiuti nell'esecuzione di una varietà di analisi al fine di estrarre conoscenze relative ai tumori.

A.15 Linea cellulare

Questa entità rappresenta esperimenti in vitro. Le linee cellulari possono essere generate da aliquote vitali mediante processi di generazione e scongelamento o da altre cellule mediante procedure di espansione.

A.16 Microarray

Rappresenta una tecnica attraverso la quale è possibile esaminare simultaneamente l'intero genoma di un organismo o la totalità dei suoi prodotti su una singola lastrina di vetro o di silicio, un chip.

A.17 Paziente

La persona la quale ha firmato un consenso informato, consentendo la raccolta di campioni di biomateriale. Sulla base di questi verranno generate aliquote e effettuati studi e analisi per scopi di ricerca.

A.18 Progetto di ricerca

I progetti di ricerca raggruppano un'insieme di attività di raccolta dei dati, sperimentazione di laboratorio e analisi dei risultati ottenuti per il raggiungimento di obiettivi comuni utili per la comunità scientifica. Al ciclo di vita di un progetto possono partecipare diversi gruppi di ricerca.

A.19 Studio

Lo studio può essere ad esempio una sperimentazione clinica o uno studio di ricerca approvato da una o più istituzioni. Ogni studio ha una serie di regole associate che definiscono i vincoli per raccogliere campioni biologici ed eseguire esperimenti di ricerca.

A.20 Vettore

Ogni entità biologica vivente richiede un contesto (ovvero un insieme di condizioni) per rimanere disponibile. Il vettore è quindi un insieme di condizioni. Allo stato attuale esistono tre principali vettori:

- H (umano): rappresenta i tessuti nel loro contesto originale (l'essere umano);
- X (trapiantato): rappresenta i tessuti quando trapiantati in un altro organismo ricevente (il topo da laboratorio);
- A S O (linea cellulare): rappresenta i tessuti quando coltivati in apposite zone di coltura (al di fuori di un organismo vivente).

A.21 Xenopaziente

[3] Rappresenta un topo da laboratorio a cui è stato compromesso il sistema immunitario e successivamente è stata impiantata un'aliquota. Diverse aliquote possono essere impiantate nella stessa cavia.

A.22 Working Group

Il WG o, tradotto in lingua italiana, gruppo di lavoro è una rete di utenti del LAS che partecipano ad attività comuni all'interno di un progetto di ricerca.

A.22.1 Principal Investigator

Il PI o, tradotto in lingua italiana, investitore principale è il ricercatore principale, nonché manager per un determinato progetto di ricerca. Tra i suoi ruoli amministrativi principali troviamo quello di formare il WG, di gestirlo e di garantire legalmente per i suoi membri durante il processo di registrazione sulla piattaforma del LAS. Può aggiungere o rimuovere utenti dai WG per il quale è responsabile e concedere o revocare l'autorizzazione ad accedere alle funzionalità per ogni utente nei propri WG.

A.22.2 Vice PI

Il vice PI è un ruolo che può essere assunto da diversi componenti di un WG. Questo incarico indica che il soggetto può, sotto determinate condizioni, sostituire momentaneamente il PI ufficiale del WG. In casi eccezionali può essere promosso a ruolo di PI.

A.22.3 Utente Semplice

Ogni utente appartenente ad un WG che non assume il né il ruolo di PI né il ruolo di vice PI.