POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

(Computer Engineering)

Tesi di Laurea Magistrale

Realizzazione di una dashboard grafica per il monitoraggio delle attività in una Biobanca dati

Relatore:
Prof. Alessandro FIORI

Candidato:
Gabriele ROLANDO
Matr. 208938

Anno Accademico 2019/2020

Un ringraziamento speciale alla mia famiglia:
è grazie al loro sostegno ed al loro incoraggiamento
se oggi sono riuscito a raggiungere questo traguardo

Indice

1 Introduzione

La rapida evoluzione tecnologica nel campo dell'oncologia biomedica e molecolare sta fornendo ai laboratori di ricerca enormi quantità di dati complessi ed eterogenei. Sono necessari sistemi automatizzati per gestire e analizzare queste conoscenze, consentendo la scoperta di nuove informazioni relative ai tumori e al miglioramento delle cure mediche [1].

Laboratory Information Management Systems (LIMS) hanno guadagnato popolarità crescente perché possono garantire buoni livelli di controllo di qualità sulle attività di laboratorio e gestire in modo efficiente le grandi quantità di dati prodotti. Questi sistemi mirano ad assistere i ricercatori nella loro pratica quotidiana di laboratorio, migliorando l'accessibilità degli strumenti e monitorando i campioni biologici e le relative informazioni.

Nell'ultimo decennio sono stati sviluppati numerosi LIMS, sia open source che proprietari. Le soluzioni commerciali sono in genere prodotti di grandi dimensioni, complessi e ricchi di funzionalità ideati per supportare facilmente grandi laboratori. I costi delle licenze possono essere proibitivi e funzionalità extra spesso comportano spese aggiuntive. I LIMS commerciali tendono ad offrire funzionalità basate sulle procedure di laboratorio più comuni, che potrebbero non adattarsi a necessità altamente specifiche. Inoltre, questi sistemi non sfruttano alcuna conoscenza di base relativa ai dati genomici e non forniscono alcuna validazione dei diversi dati genomici memorizzati nel sistema.

Molte istituzioni hanno investito nello sviluppo di soluzioni interne e/o adattato progetti open source alle proprie esigenze. Nel 2011 l'Istituto per la Ricerca e la Cura del Cancro a carattere Scientifico (IRCCS) di Candiolo, in collaborazione con il gruppo DataBase and Data Mining Group (DBDMG) del Politecnico di Torino, iniziò lo sviluppo del Laboratory Assistant Suite (LAS).

La piattaforma è un software per la gestione di una Biobanca con un'architettura modulare progettata per assistere i ricercatori in diverse attività di laboratorio. Supporta la gestione e l'integrazione di dati biomedici eterogenei e fornisce strumenti grafici per costruire analisi complesse su dati raccolti. Queste collezioni vengono definite biobanche contenenti informazioni riguardanti biomateriali catalogati e studiati in laboratorio. L'architettura modulare della piattaforma consente di gestire diversi tipi di dati non

elaborati e di tracciare dati sperimentali. Questa tipologia di struttura rende il software versatile e generico, può quindi essere esteso per supportare nuove funzionalità.

La suite è basata su un modello a più livelli, ciascuno dei quali effettua operazioni molto specifiche ed interagisce con gli altri attraverso apposite interfacce. Il livello gestionale viene utilizzato per l'autenticazione degli utenti ed il controllo degli accessi. I permessi vengono assegnati in fase di registrazione, è possibile modificarli all'occorrenza durante l'utilizzo del software. Al livello operativo si demanda la memorizzazione e il monitoraggio dei dati relativi alle diverse operazioni effettuate nei moduli. Tali operazioni sono supportate da interfacce utente studiate ad hoc per l'utilizzo in laboratorio. Il livello di integrazione è stato realizzato con lo scopo di effettuare complesse interrogazioni sui dati raccolti nei database. Infine, il livello analitico, con l'ausilio di tecnologie esterne consente agli utenti di effettuare studi approfonditi sui dati.

Ogni livello è composto da uno o più moduli, ciascuno di essi è un'applicazione web a sé stante. Per la parte di backend, ossia la parte server, si è deciso di utilizzare Django. Framework web open source e basato sul linguaggio di programmazione Python. Django è versatile e scalabile, caratteristiche indispensabili per LAS. Il frontend, la parte client, è sviluppato in HTML, CSS e JavaScript con l'ausilio della libreria jQuery. Questi linguaggi stanno alla base della programmazione web e rendono la piattaforma utilizzabile su qualunque dispositivo dotato di browser. Data la varietà di dati collezionati in laboratorio, per la persistenza degli stessi vengono impiegati due database non relazionali MongoDB e Neo4J ed il database relazionale MySQL.

L'obiettivo principale del progetto di tesi è quello di creare un'interfaccia grafica per la consultazione rapida delle principali attività non portate a termine dall'utente. Oltre a ciò, gli utenti possono visualizzare quanti xenopazienti sono in un determinato stato e/o sottoposti a trattamenti. Questo strumento deve permettere ai ricercatori di visualizzare in un'unica schermata le operazioni a loro assegnate o lasciate in sospeso ed alcune statistiche sulla distribuzione dei campioni raccolti. Inoltre, l'interfaccia dovrà consentire di riprendere i task dal punto in cui si era interrotto il lavoro.

Per la realizzazione della dashboard si è deciso in prima battuta di analizzare il funzionamento della piattaforma, nello specifico le procedure di laboratorio a cui vengono sottoposte le Aliquote, i trattamenti condotti sugli animali immunocompromessi e gli studi sulle linee cellulari. Alcune di queste operazioni necessitano di più step per concludersi mentre altre possono essere eseguite sia manualmente che mediante l'impiego di appositi robot. Per ciascuna di queste fasi, ci si è concentrati sullo studio delle transizioni di stato a cui le entità vengono sottoposte e sui passaggi fra le entità

determinati da alcune operazioni. Tutto ciò è stato rappresentato in seguito in un diagramma a stati riportante le principali transizioni di stato nonché i passaggi fra i vari tipi di entità trattate. Successivamente si è passati alla progettazione della dashboard vera e propria, individuando i dati trattati dalla nuova applicazione. Si è poi proseguito andando a disegnare la grafica per la parte di frontend con un mockup iniziale a cui è seguita la realizzazione della struttura della pagina. In seguito, si è individuato il modulo più opportuno in cui collocare la parte server della nuova applicazione web. Nel backend, in particolare, si è deciso di sviluppare delle api apposite all'interno dei moduli coinvolti per rendere la dashboard il più trasparente possibile al resto del software. Infine, sono stati individuati i casi d'uso del software per effettuarne le verifiche di operatività e di corretto funzionamento.

Il progetto di tesi è stato sviluppato partendo dall'analisi della piattaforma allo stato attuale. In prima battuta è stata effettuata una panoramica del LAS e di alcuni dei sistemi concorrenti ad oggi in commercio. Si è proseguiti con lo studio del ciclo di vita dei campioni nonché dei moduli impiegati per il trattamento delle aliquote nella piattaforma. Il capitolo successivo descrive l'analisi dal punto di vista tecnico del software, sono state studiate le tecnologie di sviluppo dei tre ambiti principali di cui è composta un'applicazione web. Pertanto, sono state analizzate le tecnologie impiegate per lo sviluppo della parte di backend e del frontend. Oltre a ciò, sono stati studiati i database adottati dalla piattaforma. Questa analisi è stata preceduta dallo studio della tecnologia impiegata per la distribuzione del software, ovvero il Docker. Si è proseguiti con lo studio del GenealogyID, codice identificativo univoco impiegato principalmente per tracciare i campioni all'interno della piattaforma. Per concludere l'analisi sono stati studiati gli stati assunti dalle aliquote in seguito alle operazioni condotte su di esse. Questo ha portato alla realizzazione di un diagramma contenente i vari passaggi di stato individuati. Sulla base di queste considerazioni è stata progettata la dashboard per poi proseguire con la realizzazione della stessa. A seguito dei test effettuati si è proceduto con il miglioramento generale della dashboard e l'implementazione di nuove funzionalità volte a facilitare l'utilizzo della pagina di riepilogo. La validità del progetto nonché il suo corretto funzionamento sono stati oggetto di valutazione mediante l'uso di appositi casi d'uso elaborati per raggiungere tale scopo.

2 Analisi della piattaforma

LAS, come anticipato nell'introduzione, è un software ideato per supportare quotidianamente i ricercatori nelle loro attività di laboratorio. Oltre a disporre delle normali funzionalità degli applicativi della sua categoria, integra dei moduli specifici per condurre analisi incrociate sui dati presenti nei database utilizzati. La piattaforma è disponibile gratuitamente su richiesta agli autori ed è interamente orientata al supporto delle attività svolte in laboratorio. L'obiettivo ultimo è la creazione di terapie nuove e più efficaci per la cura del cancro. Il LAS può essere collocato nella categoria di applicativi chiamata LIMS (Laboratory Information Management Systems). Questa tipologia di software è stata creata con lo scopo di facilitare la gestione e l'elaborazione di grandi quantità di dati, come accade quotidianamente nei laboratori.

Con l'avvento dei sistemi informatici negli anni ottanta compaiono sul mercato i primi sistemi LIMS per la raccolta dati e la gestione di alcuni flussi operativi di laboratorio. Grazie alla crescita del web, per lo sviluppo delle ultime piattaforme si è adottato un approccio web-oriented con soluzioni client-server. Questo ha portato ad una notevole riduzione dei costi e ad un incremento della facilità d'uso. Molti LIMS commerciali, per incrementare la propria appetibilità e la clientela, hanno ampliato la gamma di servizi offerti andando ad integrare funzionalità tipiche di gestionali aziendali (p.e. magazzino articoli, gestione preventivi, fatturazione ecc.). I LIMS attualmente in commercio sono quindi software gestionali a tutti gli effetti, prodotti per soddisfare i bisogni di diverse categorie di utenti quali commercialisti, studi medici in aggiunta ai laboratori.

Oltre ad applicativi commerciali, sono stati sviluppati anche LIMS open source, Bika LIMS [2] e Open-LIMS [3] ne sono due esempi. Il primo è un progetto ideato in Sud Africa e pilotato dall'industria vinicola di Western Cape, sviluppato con CMS (Content Management System) Plone [4] basato su Python. Oltre alle normali funzionalità dei LIMS, presenta dei moduli dedicati alla fatturazione ed alla gestione di documenti aziendali. Essendo open source è possibile personalizzare il codice sorgente e sviluppare in autonomia nuovi moduli. Il software originale è stato esteso nel tempo per poi venir suddiviso in diversi rami andando a creare nuovi LIMS dedicati a progetti in ambito medico ed ambientale.

Open-LIMS è un software sperimentale, non ancora completo ma disponibile in versione Alpha per l'utilizzo. Nonostante ciò, la piattaforma è tuttora molto utilizzata. Basata sul linguaggio PHP [5] e database PostgreSQL [6], può essere facilmente estesa integrando librerie PHP esterne o sviluppando nuovi moduli ad hoc. La piattaforma gestisce progetti biologici, campioni, flussi pre-sperimentali e sperimentali. Tuttavia, per effettuare l'analisi dei dati ottenuti dagli esperimenti è necessario integrare o sviluppare appositi moduli esterni.

2.1 Ciclo di vita dei campioni

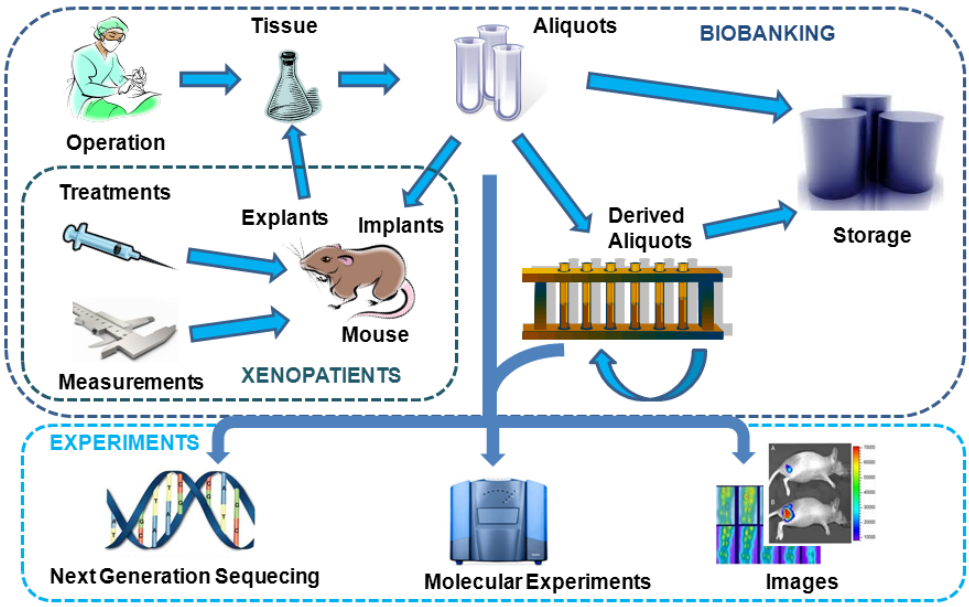

Figura 1: Schema del ciclo di vita dei campioni

I dati gestiti dalla piattaforma vengono generati da campioni di tessuti tumorali appositamente raccolti e selezionati da personale specializzato. L'ottenimento dei campioni per l'utilizzo in laboratorio è subordinato al consenso informato rilasciato dal paziente prima di essere sottoposto ad intervento chirurgico. Per la conservazione dei campioni si adottano metodi diversi in base agli esperimenti a cui sono destinati. La

conservazione in vitale, ad esempio, permette il mantenimento della vitalità del tessuto per diversi anni grazie all'utilizzo di un particolare solvente che ne impedisce la decomposizione. In ghiaccio secco il campione si conserva altresì a lungo e mantenendo intatte le componenti cellulari ma viene meno la vitalità a causa delle basse temperature a cui viene sottoposto (-80°C). Da ciascun campione è possibile dare origine ad una serie di aliquote memorizzate nella Biobanca e caratterizzate dal tipo di tumore, dal tipo di tessuto e da grandezze fisiche quali volume, concentrazione e numero di frammenti generati.

A seconda della tipologia di aliquota e del progetto di ricerca in cui essa viene impiegata, è possibile eseguire diverse attività. Con l'operazione di derivazione, ad esempio, è possibile estrarre DNA, RNA o proteine da un campione andando a creare aliquote di tipo derivato. Questa operazione può venir ripetuta sui campioni RNA estratti per ottenere campioni di tipo DNA o RNA complementare. Le aliquote derivate vengono utilizzate per condurre esperimenti molecolari e di microarray. L'impianto è un'operazione che viene eseguita su xenopazienti (topi con sistema immunitario compromesso) e consiste appunto nell'impianto di tessuto tumorale, in determinate parti del corpo dei topi. La scelta è ricaduta su topi da laboratorio in quanto presentano una struttura del DNA molto simile a quella umana. Per l'impianto vengono utilizzati campioni conservati in vitale, le cellule tumorali vengono preparate dai ricercatori per poi essere riattivate dall'organismo ospite andando a riprendere la loro attività.

Ogni progetto di ricerca prevede che vengano eseguiti determinati trattamenti sulle cellule impiantate. I ricercatori procedono quindi con l'inserimento nella piattaforma dei protocolli di trattamento oggetto di studio. Un protocollo di trattamento consiste in una o più fasi ognuna delle quali relativa ad uno specifico farmaco e di durata variabile. Per ogni fase viene definito il farmaco utilizzato, la modalità e la frequenza di somministrazione ed il dosaggio. I ricercatori, seguendo la tabella di marcia definita dal protocollo, procedono alla somministrazione dei farmaci ed al monitoraggio dello sviluppo del tumore. Sulle cellule impiantate vengono effettuate misurazioni volumetriche e di peso, queste ultime eseguite andando ad osservare le variazioni di peso dell'animale. Questi tessuti, terminati i trattamenti, possono venir espiantati, in altre parole asportati chirurgicamente, per essere studiati o per originare nuovi campioni e quindi nuove aliquote all'interno della Biobanca. Il LAS, inoltre, gestisce le linee cellulari, ovvero un insieme di campioni generati da uno stesso animale sotto le medesime condizioni di sperimentazione. Le linee cellulari vengono utilizzate nella sperimentazione in vitro e consentono di testare farmaci differenti a parità di condizioni. I ricercatori mettono a punto appositi protocolli per la generazione e per l'espansione delle linee

cellulari definendo le condizioni di sperimentazione a cui vengono sottoposte. Terminati gli esperimenti, le linee cellulari vengono archiviate, questa operazione porta alla generazione di nuove aliquote disponibili per ulteriori studi.

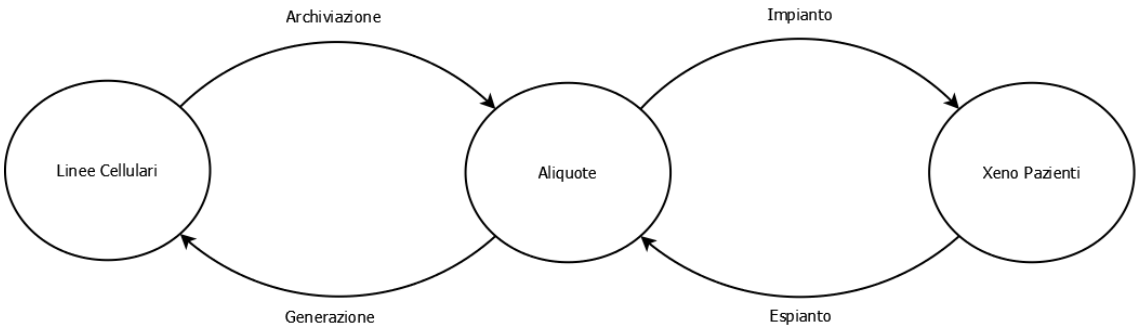

Figura 2: Sintesi delle transizioni di stato

L'insieme di queste operazioni in aggiunta agli esperimenti condotti genera una grande quantità di dati. La piattaforma mette a disposizione dei ricercatori strumenti in grado di interrogare questi dati e condurre complesse analisi. I risultati ottenuti vanno a generare nuove informazioni che migliorano la conoscenza dei tessuti tumorali oggetto di studio. Tutto ciò è finalizzato allo studio della risposta del tumore ai farmaci somministrati andando a determinare l'efficacia o meno dei trattamenti adottati.

2.2 I moduli

Il LAS presenta svariati moduli per supportare le numerose attività svolte in laboratorio. Il ricercatore viene coadiuvato dal software durante ogni fase del processo che sta eseguendo. Ogni modulo è dotato della propria interfaccia utente pensata per simulare il piano di lavoro del laboratorio dove si svolgono le funzioni per le quali il modulo è stato concepito. Per la raccolta dei campioni in laboratorio vengono utilizzati appositi contenitori. Nel modulo di Storage l'utente può gestire i contenitori comunemente presenti in laboratorio come pure inserirne di nuovi. I tubi e alcuni tipi di piastre sono destinati a contenere i campioni biologici mentre i freezer e i cassetti ospitano uno o più

sotto-contenitori. Le piastre, se formate da un insieme di tubi, possono contenere direttamente il materiale biologico. Ogni contenitore è contrassegnato da un identificativo univoco che permette ai ricercatori di conoscerne la posizione, di monitorare i campioni o i sotto contenitori al suo interno e di ottenere velocemente ciò di cui si ha bisogno. Il modulo, oltre a consentire l'inserimento di nuovi tipi di contenitori, implementa funzioni per il recupero ed il riordinamento di un contenitore.

I contenitori verranno utilizzati nel modulo della Biobanca per la raccolta dei materiali biologici. Il sistema consente l'inserimento di nuovi tessuti oppure l'archiviazione di aliquote provenienti da altri laboratori. Per la raccolta di tessuti derivanti da operazioni chirurgiche il sistema presenta l'interfaccia di inserimento di nuove aliquote dove è possibile scegliere il tipo di cellule cancerogene e le tipologie di tessuto collezionato. Nella fase successiva, per ciascun tipo di tessuto si inseriscono il tipo di conservazione e il numero di frammenti disponibili o il volume qualora l'aliquota fosse un liquido. L'archiviazione si differenzia dall'inserimento di tessuti in quanto è possibile collezionare anche aliquote di tipo derivato, ottenibili solamente attraverso opportune operazioni di derivazione in laboratorio.

La piattaforma permette di tenere traccia di una serie di **operazioni sulle aliquote**, lo svolgimento di tali operazioni è determinato dagli obiettivi del progetto di ricerca. Ogni processo di elaborazione delle aliquote prevede una fase iniziale di pianificazione ed una o più fasi di svolgimento. Inoltre, alcune operazioni possono essere svolte sia manualmente che in modalità automatica, mediante appositi macchinari presenti in laboratorio. Per lo svolgimento di queste operazioni la piattaforma prevede quindi la possibilità di scelta fra esecuzione manuale o robot. La derivazione è l'operazione con cui si possono estrarre DNA, RNA o proteine da un tessuto mediante l'uso di appositi kit e seguendo determinati protocolli operativi. Nella fase di pianificazione il ricercatore seleziona il tipo di aliquota derivata che vuole ottenere, una o più aliquote di partenza che rispettino i requisiti di derivazione e l'operatore che dovrà svolgere il processo. I requisiti sono strettamente correlati ai protocolli di derivazione, questi ultimi vengono determinati dal tipo di aliquota derivata scelto. Lo step successivo prevede la scelta del protocollo di derivazione e mostra, fra le aliquote selezionate durante la pianificazione, solo quelle compatibili con il protocollo. Nella seconda fase il ricercatore inserisce il codice a barre del kit di derivazione che verrà utilizzato. Questi due passaggi sono comuni ad entrambe le modalità di esecuzione.

Il terzo passaggio consiste nella scelta degli strumenti che verranno utilizzati, determinata dal tipo di aliquota derivata pianificato. Nell'esecuzione manuale la piattaforma presenta al ricercatore una schermata per l'inserimento dei valori di misurazione degli strumenti.

Nello step successivo il ricercatore effettua le misurazioni e inserisce i valori ottenuti. La procedura automatizzata richiede la definizione dei parametri operativi del macchinario e nella fase seguente l'inserimento dei valori restituiti. Le aliquote derivate vengono utilizzate nei moduli progettati per la gestione degli esperimenti molecolari. Operazioni quali split e rivalutazione vanno ad operare sulle aliquote derivate permettendo rispettivamente di dividere un'aliquota in una o più aliquote figlie e di effettuare nuove misurazioni di volume e concentrazione. Quest'ultima procedura prevede l'utilizzo di strumenti diversi in base al tipo di aliquota derivata e va a raffinare i dati inseriti durante la derivazione. Le operazioni di slide preparation e di slide labelling permettono di elaborare le aliquote conservate rispettivamente in formalina ed in paraffina per generare nuove aliquote utilizzate in alcuni esperimenti. La Biobanca implementa inoltre la funzionalità di trasferimento aliquote fra gli utenti attivi presenti nel database. Il sistema prevede una schermata per l'invio ed una per la ricezione delle aliquote. L'utente destinatario riceverà un'e-mail con l'avviso della disponibilità di nuove aliquote.

I campioni raccolti possono essere tessuti umani oppure **tessuti animali**. Il LAS prevede un modulo dedicato agli Xenopazienti col quale i ricercatori registrano operazioni di impianto ed espianto e tengono traccia delle misurazioni effettuate sulle cellule tumorali impiantate nei topi. Il modulo consente l'inserimento nella banca dati dei topi a disposizione nel laboratorio tracciando l'età e la provenienza. Per l'operazione di impianto il software presenta un'interfaccia che guida l'utente nei passaggi da compiere. La prima fase consiste nella scelta del protocollo da applicare, il secondo step prevede la scelta dei topi e per ognuno la scelta della parte del corpo in cui verrà effettuato l'impianto. Nella fase successiva si procede col caricamento del contenitore in cui sono presenti le aliquote conservate in vitale e la selezione dei campioni che verranno impiantati. Nell'ultimo passaggio avviene la creazione di un gruppo o l'assegnazione ad un gruppo già esistente dei topi selezionati durante la procedura.

A seguito dell'impianto, vengono effettuate misurazioni sugli animali per verificare la crescita del tessuto tumorale. Si effettuano osservazioni delle dimensioni dell'animale, misurazioni del peso, e attraverso appositi strumenti, rilevazioni del volume tumorale. La piattaforma permette di memorizzare misurazioni in 2D ed in 3D del tumore. Sono inoltre presenti apposite interfacce per misurazioni congiunte di peso e volume. I ricercatori procedono con il trattamento del tumore e la somministrazione di farmaci in base a protocolli di trattamento stabiliti in precedenza. I trattamenti prevedono diverse fasi terapiche con tempistiche di somministrazione, dosaggi e farmaci diversi. Alla loro conclusione l'utente può terminare il trattamento attraverso l'apposita schermata. Lo studio si conclude con l'espianto del tessuto tumorale che dà origine a nuove aliquote. Il

ricercatore seleziona i tessuti rimossi e procede con l'inserimento del codice identificativo del topo. La procedura è simile a quella eseguita nella Biobanca per l'inserimento di nuovi tessuti, la differenza consiste nell'assegnazione dell'identificativo univoco alle aliquote generate. La funzione che va a generare i nuovi codici utilizza come base di partenza il codice univoco dell'animale da cui derivano i tessuti. In questo modo il ricercatore può tener traccia facilmente dell'origine dei tessuti conservati.

La piattaforma è altresì progettata per supportare il ricercatore nell'esecuzione di vari **esperimenti**. Questi ultimi vengono pianificati attraverso un apposito menu presente nella home page della Biobanca. Per l'esecuzione il sistema presenta all'utente moduli sviluppati ad hoc per ciascun esperimento. Le linee cellulari, ad esempio, fanno parte degli esperimenti supportati dal sistema. Per linea cellulare si intende un insieme di aliquote generate da uno stesso paziente nelle medesime condizioni di sperimentazione. Il software è attualmente in grado di gestire la generazione e l'espansione di nuove linee cellulari e la creazione di protocolli che disciplinano le due precedenti operazioni. La generazione avviene scegliendo un protocollo fra quelli disponibili per l'operazione, nella fase successiva l'utente seleziona i tessuti conservati in vitale che andranno ad originare le linee cellulari. Nell'ultimo step si seleziona il numero di piatti creati, il tipo di linea cellulare sviluppato ed eventualmente è possibile assegnare un nickname alla collezione. L'espansione permette di generare nuove linee cellulari partendo da quelle disponibili, il ricercatore va ad inserire il fattore di diluizione utilizzato e le condizioni a cui le nuove culture sono sottoposte, se differenti da quelle originali. Inoltre, il software prevede le funzionalità di archiviazione, cancellazione e l'utilizzo di una o più linee cellulari per la sperimentazione. Attraverso la procedura di archiviazione i ricercatori generano nuove aliquote che verranno conservate nella Biobanca e saranno disponibili per nuovi test.

Nel progetto in questione, dopo aver effettuato un'analisi generale del sistema, ci si è concentrati sui seguenti moduli.

Modulo di autenticazione, dedicato alla gestione degli accessi e dei privilegi degli utenti stabiliti in fase di registrazione del profilo. I permessi utente possono venir aggiornati all'occorrenza dagli amministratori. Attualmente sono previste due tipologie di utenze oltre agli amministratori, l'utente P.I. (Principal Investigator) e l'utente LAS. Il P.I. gestisce uno o più gruppi di lavoro, ciascun gruppo viene assegnato ad un progetto di ricerca dagli amministratori. I gruppi sono composti da utenti LAS, questi ultimi sono utilizzatori del software senza privilegi particolari. Dato il suo compito di controllo, questo modulo interagisce con tutti gli altri determinando quante risorse può visualizzare un utente e a quali può effettuare l'accesso. Per questo motivo si è scelto di sviluppare la

dashboard all'interno di questo modulo andando ad interrogare opportune api degli altri moduli per ottenere i dati da visualizzare.

La Biobanca, utilizzata per la conservazione delle collezioni di campioni biologici, l'esecuzione di operazioni sulle aliquote e la pianificazione di esperimenti. Questo modulo presenta numerose operazioni che necessitano di più fasi per essere portate a termine. Si è quindi prestato maggior attenzione a come i singoli step vanno ad incidere sui dati. Queste operazioni vengono eseguite singolarmente dagli utenti e vanno ad impegnare aliquote che potrebbero essere utilizzate per altri scopi. L'obiettivo è di mostrare nella dashboard il numero esatto di procedure che necessitano dell'attenzione dell'utente affinché vengano completate nel più breve tempo possibile.

Il modulo degli Xenopazienti consente di effettuare operazioni sulle cavie del laboratorio e di misurare lo stato di avanzamento dei tumori indotti. Oltre a ciò, permette di pianificare trattamenti farmacologici utilizzati per la somministrazione dei farmaci ai topi impiantati. Traccia le operazioni chirurgiche di impianto ed espianto di tessuti tumorali nelle cavie. Il modulo interagisce con la Biobanca memorizzando i campioni espiantati e rimuovendo le aliquote utilizzate nell'impianto. I trattamenti a cui i topi sono sottoposti possono perdurare nel tempo, questo fa sì che vengano eseguiti da utenti diversi. Il sistema permette quindi ad un intero gruppo di lavoro di effettuare misurazioni sui topi operati inizialmente da un singolo membro del gruppo. Tutte le cavie da laboratorio non ancora impiantate sono invece a disposizione di qualsiasi ricercatore abbia accesso al laboratorio. Sulla base a queste regole, la dashboard mostra a tutti gli utenti il numero di topi disponibili non ancora operati e ad ogni membro di un gruppo di lavoro il numero di cavie sotto trattamento.

Il MDDM (MultiDimensional Data Manager), modulo ottimizzato per l'interrogazione dei database, consente di recuperare i dati memorizzati nei database attraverso apposite query. È possibile utilizzare query preconfigurate dagli amministratori oppure, attraverso un'apposita interfaccia grafica, consente la creazione di query anche ad utenti che non hanno studiato linguaggi di programmazione. L'interfaccia grafica mostra all'utente tre aree di lavoro, una dedicata alle entità, una agli operatori insiemistici ed una zona di lavoro centrale. Le prime due aree descritte contengono rispettivamente blocchi di tipo entità e blocchi di tipo operatore. I primi rappresentano una determinata componente, per esempio le aliquote, mentre fra i secondi troviamo alcuni dei più comuni operatori insiemistici. Per ogni blocco di tipo entità l'utente può inoltre impostare dei filtri sui dati che verranno visualizzati. Infine, l'utente può concatenare blocchi diversi nell'area di lavoro centrale per comporre una query ed estrarre le informazioni. Questo modulo espone una serie di api che lo rendono utilizzabile da tutti i moduli della piattaforma.

Nello sviluppo della dashboard si è deciso di interrogarlo per l'ottenimento delle informazioni che risultano più onerose per il sistema.

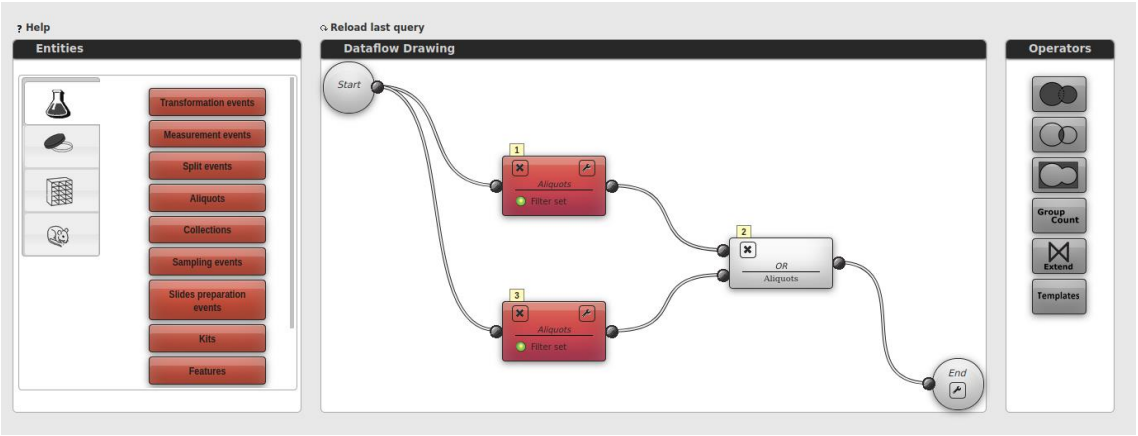

Figura 3: Interfaccia grafica del MDDM con esempio di query

3 Tecnologie

La piattaforma LAS è una web application costituita da numerosi moduli indipendenti, ciascuno di essi è dedicato ad effettuare determinate operazioni. I moduli interagiscono fra di loro e si scambiano i dati necessari per operare attraverso apposite API. Ogni componente è stato progettato seguendo le logiche di sviluppo delle applicazioni web implementando un'architettura suddivisa in tre livelli, pertanto dispongono di un'interfaccia grafica propria, un core di elaborazione ed un database dedicati.

Al primo livello viene demandata la parte grafica, il browser web sfrutta il motore di rendering in esso incorporato, per creare le pagine che vengono poi presentate all'utente. Il core applicativo, ovvero la parte server, implementa tutta la logica di business del software oltre ad elaborare ed eseguire le richieste ricevute dai client. Il terzo livello è costituito dalla base dati mediante la quale vengono memorizzate e lette le informazioni necessarie al front end ed alla piattaforma per operare.

Essendo un'applicazione destinata all'utilizzo sul web, il sistema viene reso disponibile attraverso uno o più server remoti. Ogni modulo è un'applicazione a sé stante, indipendente dalle altre, questo fa sì che il LAS possa essere impiegato su macchine separate seguendo le logiche dei sistemi distribuiti. Per definizione, un sistema distribuito è un insieme di applicazioni indipendenti che comunicano fra loro per mezzo di opportuni messaggi.

I principali vantaggi di questi sistemi sono la scalabilità, una maggior tolleranza ai guasti e la riduzione dei costi. La scalabilità è la capacità del sistema di rispondere con le stesse prestazioni all'aumentare del carico di lavoro. Ad esempio, un aumento del numero di utenti che utilizzano in contemporanea le risorse di un sistema può considerarsi un aumento del carico di lavoro. Data la presenza di più macchine è possibile predisporre contromisure che vanno a mitigare i guasti rendendo il sistema nuovamente utilizzabile del tutto o in parte. I sistemi distribuiti vanno a ridurre i costi di gestione in quanto non necessitano di grandi mainframe. Inoltre, per ciascun macchinario è possibile aggiornare le singole componenti o sostituirlo del tutto con un altro più recente, purché quest'ultimo sia in grado di utilizzare lo stesso sistema di comunicazione per interfacciarsi con il parco macchine esistente. I sistemi distribuiti presentano anche svantaggi, sono innanzitutto più complessi da realizzare e da gestire, le macchine devono poter comunicare fra loro ed è

necessario garantire la sicurezza delle comunicazioni per tutelare gli utenti. Data la grande varietà di macchine disponibili in commercio, è inoltre necessario sviluppare il software seguendo standard multipiattaforma che garantiscano la portabilità dell'intero applicativo.

3.1 Il Docker

La versione del LAS oggetto di studio fa uso della tecnologia di virtualizzazione, messa a disposizione dal Docker, per rendere fruibile il servizio e consentire la comunicazione fra i diversi moduli che lo compongono. Il Docker [7] è una piattaforma open source utilizzata per lo sviluppo e l'esecuzione di applicativi, è disponibile per i tre sistemi operativi maggiormente diffusi in commercio (Windows, Linux, MacOS). Questa tecnologia impiega appositi contenitori virtuali all'interno dei quali vengono eseguiti uno o più applicativi.

Docker nasce come software per la containerizzazione di applicazioni. Questa tecnologia semplifica la creazione di sistemi distribuiti offrendo altresì la possibilità di crearli su una singola macchina. Sono disponibili numerosi altri software progettati per adempiere allo stesso scopo. La scelta è ricaduta su Docker in quanto è un progetto open source largamente utilizzato, questo ha consentito lo sviluppo di una vasta comunità online ed il suo utilizzo per la creazione di svariati progetti. Perciò, sono stati sviluppati numerosi strumenti nonché plugin, che hanno determinato la crescita e la riuscita del progetto. Nonostante ciò, la piattaforma è leggera, non richiede risorse ingenti per il suo funzionamento. Il software è inoltre supportato da diversi sistemi operativi e recentemente anche da piattaforme cloud. Lo svantaggio principale di questa tecnologia è la possibilità di eseguire solamente un singolo processo in ogni container, per l'esecuzione in contemporanea di processi diversi su una stessa macchina sono quindi necessari più container. Nelle prime versioni del Docker si sono riscontrate alcune criticità nell'isolamento dei processi su una singola macchina, le funzioni del Kernel Linux utilizzate non incapsulano perfettamente gli applicativi. Versioni più recenti fanno uso di estensioni del Kernel dedicate alla schermatura dei processi che garantiscono una separazione più sicura fra i contenitori. Inoltre, Docker supporta contenitori creati solamente nel proprio formato, altri competitor gestiscono più formati diversi, fra i quali

il formato utilizzato dal Docker stesso, garantendo una maggior compatibilità fra gli applicativi.

Il server applicativo esegue il Docker Engine che è il software alla base della containerizzazione. Docker Engine è un'applicazione con struttura client-server, la parte server esegue il processo Docker in background ed espone una serie di API per gestire i container. Il client è un'interfaccia a linea di comando che utilizza le API per controllare ed interagire col server attraverso appositi comandi. Il processo server crea e gestisce i contenitori degli applicativi, i volumi dedicati alla persistenza dei dati ed una rete per lo scambio delle informazioni.

I contenitori comunicano fra di loro attraverso apposite interfacce di rete definite in fase di creazione, alle quali vengono assegnati indirizzi IP appartenenti alla rete generata. Docker crea i contenitori implementando alcune funzionalità proprie del kernel di Linux che consentono l'astrazione a livello virtuale di un'applicazione. Ogni contenitore utilizza, come base per l'esecuzione, il sistema operativo della macchina sulla quale viene eseguito, nel caso esso sia Linux, il contenitore farà direttamente uso del Kernel della macchina. Per macchine Windows, il sistema necessita di un'immagine intermedia per consentire ai container di accedere alle risorse del sistema operativo. Ciascuna unità predispone un ambiente con tutte le risorse e le librerie necessarie affinché l'applicazione possa essere eseguita in modo affidabile e sicuro.

L'ambiente viene configurato tramite un apposito file chiamato Dockerfile, nel quale sono presenti le istruzioni che verranno eseguite dal Docker. Questo file di configurazione contiene quindi tutti i comandi necessari a predisporre l'ambiente di esecuzione dell'applicativo, ed è necessario per effettuare l'installazione di ogni contenitore. È inoltre possibile replicare un contenitore su macchine differenti utilizzando come base di partenza lo stesso file di configurazione, in quanto l'ambiente virtuale creato al suo interno rimane invariato. Il contenitore è quindi indipendente dall'infrastruttura in cui viene impiegato, questa caratteristica viene quindi ereditata dall'applicazione eseguita al suo interno.

Queste unità, lavorando in modo autonomo, possono quindi coesistere su una stessa macchina oppure essere installate su sistemi separati. Quest'ultima caratteristica consente di semplificare la messa a punto di sistemi distribuiti. A tal proposito Docker dispone della modalità Swarm che permette di gestire container con relative applicazioni distribuiti su più macchine. Le ultime versioni del software supportano nativamente la modalità Swarm, attraverso apposite istruzioni su linea di comando si può creare uno Swarm, gestirne il comportamento e la messa in funzione di un'applicazione.

Per la creazione di uno **Swarm**, ogni macchina facente parte del pool deve avere un'istanza del Docker attiva ed essere collegata in rete. Le comunicazioni sono protette imponendo agli host nella rete l'uso di TLS con mutua autenticazione. A seconda della configurazione impostata, ciascun calcolatore può svolgere il ruolo di manager dello Swarm, il ruolo di operatore o entrambi contemporaneamente. I processi che partecipano alla rete di Swarm prendono il nome di nodi. Un calcolatore può ospitare più nodi appartenenti allo stesso Swarm o a Swarm differenti. Il nodo manager invia i task ai nodi operatori, a sua volta può ricoprire il ruolo di operatore, ricevere task da parte di altri manager ed eseguirli. Inoltre, al manager spettano i compiti di coordinare e di gestire gli operatori appartenenti alla sua rete per mantenerne ottimale lo stato. L'aggiornamento di un nodo può essere effettuato in modo incrementale, creando un nuovo nodo col software aggiornato ed effettuando un reindirizzamento al nuovo nodo. Qualora la nuova versione presentasse errori è possibile ripristinare il nodo precedente per effettuare le opportune correzioni e al contempo garantire l'erogazione del servizio.

Il Docker dispone inoltre di **Compose**, strumento che permette l'esecuzione di applicazioni facenti uso di svariati contenitori. Questo strumento utilizza un apposito file di configurazione di tipo YAML contenente l'elenco dei contenitori e dei volumi di cui l'applicativo necessita. YAML è un formato per la memorizzazione di dati facilmente utilizzabile da esseri umani e macchine. Questo file, al pari dei contenitori da esso generati, è utilizzabile su qualunque altra macchina su cui sia installato Compose. Nel file vengono inoltre memorizzate le informazioni necessarie alla creazione del sistema distribuito. Per ogni contenitore vengono specificati il nome, le porte utilizzate per la comunicazione ed eventuali dipendenze da altri contenitori, quali ad esempio le unità destinate ad ospitare i database. Per la preparazione dell'ambiente di esecuzione viene lanciato il comando di compilazione del file di configurazione. Questa operazione elabora il file andando a creare i container ed i volumi in esso specificati, successivamente vengono processati i singoli Dockerfile per l'installazione degli applicativi nei rispettivi container. Al termine della procedura, è possibile lanciare il comando per l'esecuzione vera e propria dell'applicazione e di tutti i servizi ad essa collegati precedentemente definiti. Compose permette quindi di creare un sistema distribuito virtuale utilizzando un singolo host fisico, ogni contenitore può essere visto come un processo a sé stante.

Per l'installazione del LAS si è quindi deciso di utilizzare il Docker, software che offre una maggior flessibilità di configurazione alle organizzazioni interessate ad utilizzare la piattaforma. Nella versione oggetto di studio viene impiegato Compose per installare ed eseguire l'applicativo. Il progetto dispone di un file di configurazione globale di tipo

YAML contenente le istruzioni necessarie al Docker per la creazione dei container e dei volumi impiegati dal software.

Al suo interno sono presenti i dati relativi al contenitore nel quale viene eseguito il LAS, un volume ed un contenitore per ciascun database. È inoltre presente un contenitore di servizio per semplificare la lettura dei dati presenti nel database SQL da parte degli amministratori. Per ciascun database, oltre al contenitore ed al volume ad esso riservati, viene specificata l'immagine con la versione della base dati che verrà utilizzata. Ogni unità impiega un Dockerfile le cui istruzioni permettono al Docker di effettuare la configurazione dell'ambiente di esecuzione. Nella definizione del container che ospiterà la piattaforma sono elencate le dipendenze per il collegamento ai database e le porte attraverso le quali il server sarà in ascolto.

Durante la fase di compilazione della piattaforma vengono eseguite le istruzioni presenti nel file di configurazione globale per la creazione dei container, successivamente il sistema elabora ciascun Dockerfile e prepara gli ambienti. Nel caso specifico dei database, viene scaricato il relativo software dalla libreria di Docker per poi essere installato nel contenitore, tutto ciò elaborando l'immagine precedentemente descritta. Docker dispone di una libreria online contenente Dockerfile opportunamente compilati per l'installazione dei più comuni database. In questa fase iniziale, i database vengono altresì popolati con i dati preliminari necessari alla piattaforma per operare, queste informazioni ed i successivi dati immessi dagli utenti sono memorizzati nei relativi volumi. Future compilazioni o esecuzioni dell'applicativo non vanno ad intaccare i volumi ed i loro contenuti, per rimuoverli è necessario utilizzare un comando apposito. Questa è una delle caratteristiche principali di Compose volta ad evitare un'eventuale perdita di dati dovuta a nuove compilazioni durante la fase di sviluppo del software e principalmente quando verrà utilizzato dagli utenti finali.

Per l'esecuzione della piattaforma viene consigliato l'utilizzo di una macchina con sistema operativo Linux al fine di facilitare l'installazione e la messa in funzione del sistema. Durante la configurazione del contenitore in cui viene eseguito il LAS vengono installate tutte le dipendenze elencate nel Dockerfile, esaminando il file si nota come il container richieda il sistema operativo Ubuntu nonché il server HTTP Apache [8]. Nel file sono inoltre presenti tutte le librerie necessarie al server Apache per operare oltre a quelle destinate all'esecuzione di codice Python. Infine, sono presenti le impostazioni con cui Apache fornirà i servizi, ovvero le porte esposte agli utenti ed il certificato SSL per rendere più sicure le comunicazioni. Il Dockerfile viene nuovamente elaborato quando viene lanciato il comando di esecuzione del software. In questa fase viene avviato l'applicativo ed i servizi ad esso collegati, generalmente non si effettua una nuova

compilazione. Questo accade qualora venga riscontrata la mancanza di uno o più contenitori oppure l'operazione venga forzata dall'amministratore attraverso apposite opzioni aggiunte al comando. Al termine dell'operazione, il servizio risulta disponibile per la fruizione.

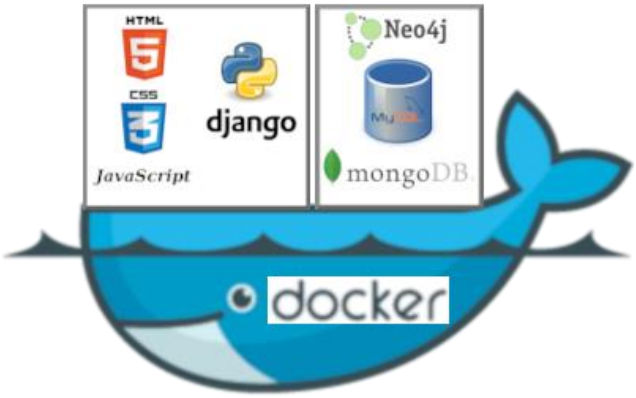

Figura 4: Struttura della piattaforma

Gli autori del LAS consigliano di configurare il Docker in modalità Swarm impiegando almeno due macchine per poi procedere opportunamente all'installazione dei moduli del software. Una macchina viene riservata allo storage, andrà quindi ad ospitare i database ed i servizi ad essi collegati. Il secondo server è dedicato ai moduli dell'applicativo vero e proprio ed esporrà il software agli utenti finali. Le prestazioni ed il numero dei calcolatori impiegati dipendono da svariati fattori, i principali sono il carico di lavoro calcolato in termini di utenti collegati simultaneamente ed il numero di entità biologiche che presumibilmente verranno collezionate dal sistema. Questa configurazione è adatta a laboratori di piccole-medie dimensioni, per strutture di maggior grandezza è consigliato l'impiego di un numero più alto di macchine per garantire una miglior suddivisione del carico di lavoro.

3.2 Backend

Il backend, ovvero la parte server di un'applicazione, contiene la logica di business dall'applicativo, rappresenta quindi il core di elaborazione delle informazioni e non è visibile all'utente finale. Questa parte del software si occupa generalmente di fornire l'applicativo al client per poi elaborare e rispondere opportunamente alle richieste ricevute. Inoltre, è responsabile della sicurezza dei dati memorizzati nei database e dell'autenticazione degli utenti che accedono al software. La piattaforma LAS è pensata per essere utilizzata sia in una Intranet, come può essere una rete interna ad un istituto di ricerca, sia online attraverso Internet. In entrambi i casi d'impiego, il backend impone l'utilizzo del protocollo HTTPS per garantire che le comunicazioni avvengano in modo sicuro fra il server applicativo e le macchine degli utenti. Viene utilizzata un'architettura di tipo client-server, un utente si connette, attraverso un browser web che funge da client, all'applicazione messa a disposizione dal server remoto. Il protocollo HTTPS, in prima istanza, impone che venga creata una connessione sicura fra le due macchine. Al termine di questa procedura, il client scarica la parte di applicazione ad esso riservata e l'utente può iniziare ad utilizzare il software.

Per lo sviluppo della parte server di un applicativo web sono disponibili numerosi linguaggi, ognuno dei quali offre soluzioni differenti. Molte di queste sono basate su framework, sviluppatisi nel tempo in seguito alla sempre maggior richiesta di applicazioni fruibili su browser.

Java [9], linguaggio ad alto livello orientato agli oggetti, originariamente adottato per lo sviluppo di applicazioni desktop indipendenti dall'hardware su cui vengono eseguite. Questo perché per l'esecuzione è necessario utilizzare un interprete del linguaggio, generalmente una macchina virtuale, che converte il codice in istruzioni eseguibili dalla macchina. Il codice deve quindi essere elaborato due volte prima di essere eseguito, viene inizialmente convertito in Java bytecode, linguaggio intermedio fra il linguaggio macchina ed il linguaggio di programmazione. Successivamente, la macchina virtuale interpreta il bytecode e lo converte in linguaggio macchina. In ambito web vengono utilizzati appositi Application Server che implementano le specifiche Java e ne permettono l'esecuzione. Il linguaggio dispone di numerose librerie ed è alla base di diversi framework, il più noto nell'ambito web è Spring. Nonostante ciò, le applicazioni in Java risultano più complesse in quanto è necessario effettuare diverse operazioni al fine di rendere l'applicazione operativa.

PHP, linguaggio di scripting open source nato appositamente per sviluppare il backend web. È multipiattaforma, può quindi essere impiegato sia in ambienti basati su Unix (Linux e MacOS) sia su Windows. Linguaggio fra i più utilizzati data la sua semplicità e la velocità con cui permette di creare un sito web partendo da zero. Grazie a queste caratteristiche ed alla popolarità acquisita, sono stati creati numerosi framework per semplificare ulteriormente lo sviluppo. Inoltre, su di esso si basano molti Content Management Systems (CMS), software pensati per permettere ad un utente che non ha conoscenze di programmazione di produrre un sito web. È un linguaggio interpretato, le istruzioni vengono convertite direttamente in linguaggio macchina, non è necessario impiegare una macchina virtuale. Quest'ultima caratteristica influisce direttamente sulle performance del sistema, un codice molto complesso o una quantità elevata di dati processati andranno a ripercuotersi negativamente sulle prestazioni complessive del sistema.

Python [10], uno dei linguaggi che si sta affermando maggiormente lato backend negli ultimi anni. Questo successo è dovuto principalmente alla nascita di due framework, Django [11] e Pyramid [12], progettati appositamente per il web. Python è un linguaggio orientato agli oggetti che supporta la definizione di classi ed il concetto di ereditarietà delle classi. Il codice fa uso di una sintassi molto essenziale, inoltre va ad utilizzare l'indentazione delle righe per delimitare blocchi logici. Questo rende il codice semplice e facilmente leggibile. Gli sviluppatori hanno creato un'ampia libreria standard che va a gestire e semplificare numerose operazioni quali, ad esempio, la lettura e la scrittura di file. Python è inoltre open source e può essere eseguito su tutti i più comuni sistemi operativi ad oggi in commercio.

JavaScript, linguaggio orientato agli oggetti creato originariamente per rendere dinamiche le pagine web tramite l'uso di script eseguiti sul client. Il linguaggio è orientato agli oggetti ed è interpretato, non necessita quindi di essere compilato. Inoltre, il codice è debolmente tipizzato, non esiste un meccanismo di controllo del tipo di variabile che si sta utilizzando, è quindi possibile assegnare tipi diversi (intero o stringa) ad una stessa variabile in punti diversi del codice. Il linguaggio ad oggi è utilizzato sia lato client, interpretato dal browser dell'utente, sia lato server dove viene impiegato in framework che necessitano di compilare il codice JavaScript ma ottengono prestazioni elevate in esecuzione. Recentemente utilizzato dal framework Node.js [13] con un approccio orientato agli eventi. Un server che utilizza Node.js rimane inattivo in attesa delle chiamate da parte dei client. Per ogni connessione ricevuta viene lanciato un evento ed eseguita, in modo asincrono, la funzione ad esso dedicata. L'accesso alle risorse del sistema da parte di ogni processo risulta quindi limitato nel tempo. Eseguendo in maniera

asincrona, qualora un processo debba effettuare operazioni di input/output, il sistema non rimane bloccato ma può servire un altro client. Node.js non è adatto ad eseguire processi che fanno un alto uso della CPU, che andrebbero quindi a generare dei colli di bottiglia. Questo è dovuto al modo in cui viene eseguito JavaScript, essendo un linguaggio destinato al front end viene processato da un singolo thread per operazioni veloci, che non richiedono un utilizzo intensivo del processore. Per ovviare a questo problema, nelle versioni più recenti di Node.js è stato introdotto il multithreading.

In seguito ad un'attenta analisi, gli sviluppatori della piattaforma LAS hanno scelto di implementare il backend in **Python**, impiegando il framework dedicato al web Django. Il linguaggio è open source, facilmente leggibile grazie alla sintassi ridotta ed è versatile. La versatilità è dovuta al fatto che è un linguaggio interpretato e di alto livello. Per linguaggio interpretato si intende un linguaggio che non necessita di essere compilato, il codice viene eseguito direttamente su qualunque macchina. I linguaggi di alto livello sono tutti quei linguaggi che permettono al programmatore di concentrarsi esclusivamente sulla logica del software che verrà prodotto, le funzioni ad alto livello utilizzate 'nascondono' una serie di operazioni che vengono eseguite per conto del programmatore senza necessità che quest'ultimo le implementi. Le ridotte regole sintattiche facilitano altresì l'apprendimento del linguaggio da parte dei neofiti, riducendo il tempo necessario per diventare produttivi. Dispone di una libreria standard ricca di funzioni volte a facilitare e velocizzare il compito degli sviluppatori implementando soluzioni precostituite e aumentando l'efficienza del codice.

Il linguaggio è dinamicamente tipizzato, ovvero il tipo di una variabile viene controllato a runtime e può cambiare nel corso dell'esecuzione. Sono quindi supportati diversi paradigmi di programmazione incrementando gli ambiti in cui è possibile utilizzare il linguaggio. Questa caratteristica si contrappone alla tipizzazione statica che prevede l'assegnazione di un tipo ad ogni variabile ed il controllo in fase di compilazione. Lo svantaggio principale è legato alla velocità di esecuzione, il codice viene interpretato direttamente dal processore in fase di esecuzione, ciò comporta un carico di lavoro maggiore in quanto ogni istruzione deve essere convertita in linguaggio macchina e poi eseguita. La tipizzazione dinamica può portare ad errori che vengono riscontrati solamente in fase di esecuzione e che spesso necessitano di un tempo lungo di debug per essere individuati. Considerato l'ambito di utilizzo del LAS, ovvero come strumento a supporto in un laboratorio di ricerca, gli svantaggi riscontrati nell'impiego di Python sono trascurabili rispetto ai vantaggi precedentemente elencati. La gestione dinamica dei tipi delle variabili permette altresì di operare agevolmente con i dati eterogenei trattati in laboratorio. Tutto ciò è coadiuvato dall'impiego del framework Django, pensato

appositamente per facilitare lo sviluppo di applicazioni web. Django è un framework open source che impiega la vasta libreria standard offerta da Python per velocizzare lo sviluppo di nuove applicazioni. Il framework fornisce soluzioni pronte all'uso legate alla gestione della sessione utente ed agli accessi ai database. Inoltre, sono implementati accorgimenti automatici per gli aspetti più comuni legati alla sicurezza in ambito web quali la prevenzione di attacchi di SQL injection e CRSF (Cross-site Request Forgery). Viene così velocizzato lo sviluppo senza comprometterne la sicurezza.

Ogni applicazione basata su Django utilizza un file di configurazione che contiene le impostazioni del progetto. In questo file sono presenti le configurazioni per la connessione ai database impiegati, per la sicurezza dell'applicazione oltre ad impostazioni legate all'autenticazione degli utenti e le sessioni di accesso. Sono inoltre presenti configurazioni per il funzionamento del server legate all'invio di e-mail, all'impiego della cache ed alla gestione dell'upload di file da parte dei client. Il framework è progettato per consentire uno sviluppo full stack di un'applicazione web. Lato server fornisce una netta separazione fra il livello che interagisce con il database ed il livello applicativo rendendo il software scalabile, requisito indispensabile per un sistema distribuito. Supporta nativamente i più comuni database relazionali, in seguito sono state sviluppate dalla comunità online, librerie ad hoc per la gestione dei database non relazionali. Nella parte client risiede il codice HTML che compone le pagine web, queste vengono rese dinamiche anche mediante l'uso di appositi tag che racchiudono funzioni Python. È inoltre possibile integrare i più comuni framework JavaScript di sviluppo front end ad oggi disponibili, questo per separare ulteriormente la parte server da quella client. Django è ampiamente utilizzato, in rete sono presenti numerose librerie di terze parti testate ed affinate dalla comunità di sviluppatori. Inoltre, il framework fornisce nativamente librerie per la gestione dei dati inseriti dagli utenti mediante l'uso di form preconfigurati e per la creazione di pagine web multilingue.

Il framework utilizza il paradigma di programmazione **MVT** (Model-View-Template), una versione rivisitata del più comune MVC (Model-View-Controller). Il Model è un file Python che contiene la definizione della classe corrispondente all'entità trattata, la classe dispone di tutti i campi necessari a descrivere l'entità di riferimento. Generalmente viene impiegato un Model per ogni tabella presente nel database. Nel progetto LAS si è optato per l'utilizzo di un unico Model contenente tutte le definizioni delle classi necessarie al modulo per poter operare. Ogni modulo dispone quindi di un singolo file Model nel quale viene mappata la struttura completa del database relativo. La libreria dedicata ai Model semplifica la gestione di tutte le operazioni essenziali sui dati nel database. Attraverso una singola funzione Python, il framework gestisce per conto del programmatore

l'apertura e la chiusura della connessione al database nonché la creazione e l'esecuzione della query corrispondente all'operazione.

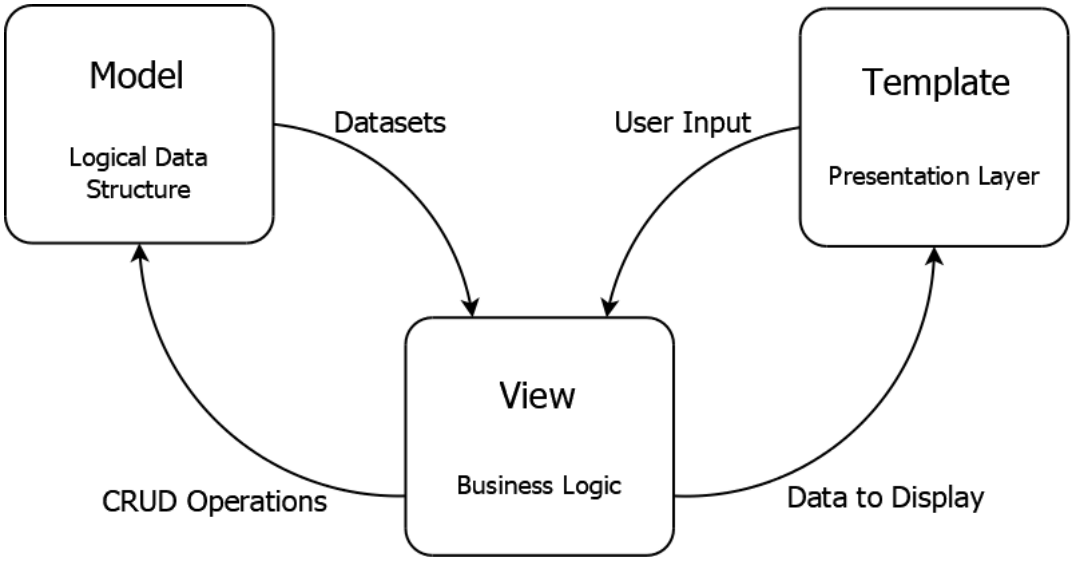

Figura 5: Schema Model View Template

Il Template rappresenta l'interfaccia grafica, opportunamente popolata con i dati e restituita al browser dell'utente sottoforma di risposta HTTP. Il LAS utilizza il concetto di template generico messo a disposizione da Django, ossia un template riutilizzabile più volte per mostrare dati diversi ma appartenenti ad una stessa tipologia. Il concetto viene applicato altresì alle viste statiche, la navbar ed il footer ad esempio, fanno parte di un template 'base' che viene esteso dagli altri template del modulo. Nel file di configurazione viene specificato l'engine da utilizzare per la gestione dei Template, Django dispone nativamente di due diversi engine per i Template e consente la creazione di uno ad hoc qualora necessario. Inoltre, sono indicati i percorsi nel file system dove il motore di rendering può trovare i Template per processarli.

La View svolge il ruolo di Controller implementando il core di elaborazione che rende funzionante l'applicazione. Elabora le richieste web ricevute effettuando, qualora necessario, operazioni sui database attraverso i Model. A seconda del risultato ottenuto, richiama opportunamente il Template che verrà inviato come risposta web agli utenti. Il framework sviluppa una generica applicazione web nel seguente modo: ogni View contiene una o più funzioni che eseguono una parte della logica di business

dell'applicazione. Ciascuna funzione richiama uno o più Template in base alle informazioni richieste ed al risultato ottenuto dall'elaborazione. Le funzioni nelle View sono raggiungibili attraverso determinati URL, Django tiene traccia di ogni indirizzo mappando ciascuna View al rispettivo URL. Il framework consente di gestire, ed eventualmente bloccare, gli accessi alle funzioni mediante appositi decoratori. Ad esempio, sono presenti decoratori per limitare l'accesso alla funzione attraverso i soli metodi GET e POST, altri vengono utilizzati per consentire l'accesso esclusivamente agli utenti che hanno effettuato il login.

3.3 Frontend

Con il termine frontend si intende l'interfaccia grafica dell'applicativo sviluppato, la quale viene mostrata all'utente finale e gli permette di usufruire del servizio. Può essere utilizzato sia in ambito web che in ambito mobile in quanto anche le app per smartphone necessitano di un'interfaccia utente per poter essere utilizzate. Questa parte del software ha acquisito col tempo un'importanza crescente in seguito all'introduzione della vendita online rappresentando quindi una vera e propria vetrina virtuale per i negozi di e-commerce. Questo ed altri fattori hanno portato alla nascita di framework dedicati esclusivamente allo sviluppo della parte client di un'applicazione. I più recenti vengono impiegati per la definizione di interfacce fruibili sia sul web che in applicazioni mobile, adottando opportuni accorgimenti per la gestione dei diversi schermi dei dispositivi.

Per lo sviluppo del frontend sono quindi disponibili molti framework con caratteristiche differenti per soddisfare la crescente richiesta di applicazioni web negli ambiti più svariati. La maggior parte di essi è basata su JavaScript, linguaggio di scripting creato appositamente per il frontend.

React [14], libreria JavaScript open source che implementa il concetto di DOM virtuale (Document Object Model). Ogni browser crea un oggetto DOM nel momento in cui la pagina web viene caricata per essere mostrata all'utente. Il DOM HTML è un modello contenente tutti gli elementi HTML presenti nella pagina, ciascun elemento è un oggetto del DOM ed è manipolabile mediante funzioni JavaScript. React crea un oggetto DOM virtuale in corrispondenza di ciascun DOM reale che verrà elaborato dal browser. L'oggetto virtuale ha le stesse proprietà di quello reale ma non porta a modifiche visibili

da parte dell'utente. Questo consente di manipolare l'oggetto virtuale molto velocemente rispetto alle modifiche apportate al DOM reale in quanto non viene mostrato nulla. React opererà successivamente sul DOM reale andando a modificare solamente le parti che hanno subito variazioni e lasciando inalterato il resto dell'oggetto. Per far ciò, vengono utilizzati due oggetti virtuali, il primo contiene le modifiche apportate dall'operazione eseguita, mentre il secondo rappresenta una 'fotografia' dell'attuale DOM mostrato all'utente. React effettua una comparazione fra i due oggetti e modifica l'oggetto reale basandosi sulle differenze riscontrate.

La libreria è pensata per creare una struttura a componenti, il programmatore può dividere la pagina web in uno o più componenti, anche annidati fra loro. Ciascun componente può essere riutilizzato in qualunque parte dell'applicazione. Inoltre, React permette di utilizzare una sintassi chiamata JSX (JavaScript XML) per descrivere gli elementi del DOM. Questa sintassi è un'estensione del JavaScript e consente di inserire elementi HTML nella pagina senza utilizzare le funzioni JavaScript dedicate (createElement o appendChild). In JSX è quindi possibile scrivere espressioni che contengono tag HTML e funzioni JavaScript al loro interno, durante la compilazione le espressioni verranno convertite in normali funzioni JavaScript. Date le sue caratteristiche, il framework viene utilizzato principalmente per creare SPA (Single-Page Applications), inoltre, risulta leggero per il browser, la compilazione produce un pacchetto di dimensioni ridotte. Richiede un'ottima conoscenza del JavaScript per utilizzare al meglio le espressioni JSX, questo lo rende non semplice da imparare in breve tempo. Essendo pensato per applicazioni a continuo aggiornamento dei propri contenuti, non è utile per pagine web pressoché statiche. Il rapporto fra costo di apprendimento e tempi di sviluppo di queste ultime non è ottimale.

Angular [15], framework basato su TypeScript [16] per la creazione di applicazioni a pagina singola (SPA). TypeScript è un linguaggio che estende JavaScript, perciò qualunque script in JavaScript può essere inserito ed elaborato direttamente nel codice TypeScript. Angular utilizza una struttura basata su moduli, ogni modulo contiene al suo interno uno o più componenti che possono essere riutilizzati anche in moduli diversi da quello di origine. Sia i moduli che i componenti sono classi TypeScript con appositi decoratori impiegati per la loro distinzione. I componenti definiscono le pagine web che verranno mostrate all'utente, generalmente una pagina è composta da diversi componenti annidati, ciascuno dei quali dedicato ad una parte o ad una funzionalità specifica. Ogni componente è rappresentato da una classe al cui interno risiede la logica applicativa ed è associato ad un template HTML che incapsula ciò che verrà mostrato all'utente.

La caratteristica peculiare di Angular è conosciuta come two-way data binding, ossia il meccanismo che lega un elemento dell'interfaccia grafica con il modello dei dati e viceversa. Questo meccanismo sincronizza automaticamente ciò che viene mostrato all'utente con il modello dei dati presente in background. Questa caratteristica consente di creare pagine con elevata interattività ma influisce sulle prestazioni complessive dell'applicazione. Perciò, la funzionalità non è sempre attiva, come nella prima versione del framework, ma è possibile utilizzarla solo nei punti di codice in cui è necessaria. Per l'utilizzo di questa ed altre funzionalità, sono disponibili numerose librerie che vengono importate nel progetto. Tutto ciò incrementa la complessità degli applicativi e li rende più pesanti per l'esecuzione su browser. Inoltre, la struttura di un progetto Angular può essere molto dinamica, consente di creare la stessa pagina in molti modi diversi. Ciò rende l'apprendimento del framework non banale. Dispone però di una vasta comunità di sviluppatori nonché di molta documentazione che semplificano le scelte di sviluppo.

Vue.js [17] framework JavaScript leggero e flessibile, ideato per poter essere adottato in modo incrementale in applicazioni già esistenti. Il framework si presta quindi come base sia per la creazione di intere applicazioni, sia per lo sviluppo di un'estensione di un'applicazione già esistente. Questa seconda opzione è resa possibile dal fatto che Vue.js è anche disponibile come libreria per essere importata in una singola pagina web. Il framework è stato ideato e sviluppato in tempi recenti rispetto ai due analizzati in precedenza. Gli sviluppatori hanno quindi tenuto in considerazione i punti di forza e gli svantaggi dei due framework concorrenti durante la realizzazione. Vue.js implementa quindi il concetto di DOM virtuale e adotta una struttura a componenti. Inoltre, consente l'uso del two-way data binding così come della notazione JSX. Essendo stato realizzato di recente, non dispone ancora di una comunità al pari dei concorrenti, inoltre, alcune librerie o funzionalità non sono ancora perfettamente affidabili.

jQuery [18], libreria JavaScript open source, nata con lo scopo di ridurre la quantità di codice necessario per eseguire operazioni molto comuni e che devono essere ripetute spesso. La libreria mette a disposizione dei metodi, per eseguire una vasta gamma di operazioni, che racchiudono al loro interno molte istruzioni JavaScript ottimizzate allo scopo. Lo sviluppatore può quindi utilizzare uno o più di questi metodi evitando di riscrivere le stesse righe di codice più volte. Vengono quindi semplificate le operazioni legate alla gestione degli eventi, quali click del mouse o input da tastiera, le chiamate HTTP per la comunicazione dei dati al server così come l'aggiunta e la rimozione di elementi nel DOM.

Uno dei punti di forza della libreria è l'API dedicata alla gestione delle chiamate asincrone inviate al server. Le chiamate asincrone sono indispensabili per evitare

rallentamenti o blocchi dell'interfaccia grafica mentre si attende la risposta dal server. La libreria risulta quindi ricca di funzionalità, questo comporta un vantaggio per lo sviluppatore ma uno svantaggio per l'utente finale in quanto deve essere scaricata interamente dal browser web. Inoltre, alcune operazioni vengono implementate automaticamente dai browser più moderni rendendo la libreria di fatto obsoleta. Nonostante questi aspetti negativi, è tuttora molto utilizzata per la facilità e la rapidità di sviluppo che ne deriva nonché per la quasi totale compatibilità con i browser, soprattutto con quelli più datati ma tuttora ancora impiegati in ambiti legacy. È quindi una valida alternativa all'impiego dei framework più recenti, descritti in precedenza, per lo sviluppo di tutte quelle applicazioni che non necessitano di operare come SAP e richiedono tempi di sviluppo brevi e sistemi ampiamente collaudati.

Nella versione oggetto di studio, gli sviluppatori hanno utilizzato quanto offerto da **Django** in merito alla gestione dei Template HTML, integrando alcune funzionalità mediante l'impiego di jQuery. Questa scelta è legata a ristrette tempistiche di sviluppo e garantisce la massima stabilità nonché compatibilità da parte di tutta l'applicazione. La parte client del LAS è quindi costituita dall'insieme di tutti i Template presenti nei vari moduli. I creatori del framework Django hanno realizzato un linguaggio ad hoc per la composizione dei Template chiamato Django template language. Questo linguaggio può essere utilizzato per creare file testuali in qualunque formato (HTML, CSV, XML, ...) ed è basato sull'utilizzo di variabili popolate dal backend e di appositi tag per il controllo della logica. In seguito, le variabili vengono sostituite con il loro valore quando il template viene processato. Il linguaggio dispone inoltre di filtri per la formattazione del contenuto delle variabili. Questi filtri vengono applicati al contenuto della variabile solamente in fase di visualizzazione del template mantenendo inalterato il contenuto della stessa.

L'engine per la gestione dei template in Django implementa il concetto di ereditarietà dei template. Questo principio consente la creazione di un template base contenente lo scheletro della pagina composto da diversi blocchi. Il template base viene successivamente esteso dai template figli che andranno ad effettuare un override dei blocchi. Utilizzando questo concetto, chiamato anche template generico, diminuisce la quantità di codice ridondante ma si va ad incrementare il numero complessivo di file. La manutenzione ed eventuali modifiche al sistema risultano quindi facilitate a discapito dello spazio occupato sul disco. Nel progetto oggetto di studio, i Template sono file HTML al cui interno vengono inseriti appositi tag per rendere dinamico il contenuto della pagina. Utilizzando i tag si possono svolgere molteplici compiti quali, ad esempio inserire iterazioni o costrutti condizionali, eseguire il debug, estendere altri template e richiamare

informazioni esterne per aggiornare la pagina corrente. Ogni Template è una pagina realizzata ad hoc per ciascuna schermata o funzionalità dell'applicazione.

Per impaginare correttamente gli elementi HTML all'interno della schermata, si è optato per l'utilizzo del **CSS** (Cascading Style Sheets). Questo linguaggio di stile definisce come il browser dell'utente mostrerà i contenuti della pagina. Gli stili ed il tema di ogni pagina sono forniti attraverso opportuni modificatori di proprietà quali ad esempio, la posizione ed i colori, presenti nei file CSS. I modificatori vengono applicati agli elementi HTML attraverso opportuni identificativi, i più comuni prevedono l'utilizzo diretto dei tag HTML o degli attributi *id* e *class*, presenti in ogni elemento. Ciascun foglio di stile CSS è quindi una lista di identificativi al cui interno vengono inseriti i modificatori di proprietà. Inoltre, la piattaforma fa uso della libreria grafica jQuery UI [19]. Questa libreria è stata sviluppata come estensione di jQuery con lo scopo di rendere ancora più interattive le pagine web. Offre una serie di widgets ed effetti grafici preconfigurati e pronti all'uso, ne permette la modifica a livello grafico al fine di mantenere lo stesso stile in tutta l'applicazione.

Il **JavaScript** viene impiegato per scrivere funzioni che eseguono determinate azioni allo scatenarsi di specifici eventi, come il click su un pulsante. Questo al fine di rendere la pagina web interattiva in risposta agli input forniti dall'utente. Sono disponibili in commercio numerose librerie, molte di queste open source, che vanno ad integrare e semplificare l'utilizzo di questo linguaggio. Nel progetto LAS, JavaScript è utilizzato esclusivamente lato client, viene quindi eseguito direttamente sul browser dell'utente. Il linguaggio rende le pagine dinamiche andando a gestire gli eventi generati dall'utente, nello specifico vengono gestiti i click effettuati sui pulsanti presenti nella piattaforma. Come anticipato, il progetto utilizza la libreria jQuery andando ad impiegare, fra le altre, le API dedicate a stabilire connessioni asincrone con il server. Queste chiamate vengono effettuate mediante costrutto AJAX (Asynchronous JavaScript And XML) che consente di aggiornare solo alcune parti della pagina corrente senza effettuare un aggiornamento completo. jQuery mette a disposizione una serie di funzioni dedicate alle chiamate AJAX al cui interno includono callback per la gestione dei risultati o degli errori ricevuti in risposta. La libreria implementa funzioni sia ad alto livello, preconfigurate per effettuare chiamate specifiche, sia di basso livello, in cui il programmatore ha un controllo maggiore sui parametri delle chiamate che verranno effettuate.

3.4 Database

La memorizzazione dei dati e la loro successiva lettura sono aspetti fondamentali per qualunque applicazione. I database sono quindi stati creati con lo scopo di facilitare e velocizzare queste operazioni. Questi sistemi consentono di immagazzinare, organizzare e gestire grandi quantità di informazioni mediante l'uso di un unico software. Vengono suddivisi principalmente in due categorie distinte, i database relazionali e i database NoSQL. I database relazionali sono basati sul linguaggio di programmazione SQL (Structured Query Language), utilizzato per compiere operazioni sui dati. Le informazioni vengono memorizzate in tabelle, la cui struttura è decisa in fase di progettazione del software. Le tabelle sono composte da colonne, impiegate per la definizione dei differenti campi, e da righe contenenti i record con i dati corrispondenti a ciascun campo. Questo tipo di struttura rende la base dati poco flessibile, future modifiche risultano generalmente onerose per il sistema.

I software per la creazione e la gestione di questa tipologia di database prendono il nome di **Relational Database Management Systems** (RDBMS). In commercio, troviamo numerosi RDBMS, alcuni dei quali impiegano una versione di linguaggio SQL con modifiche ad hoc per sopperire a determinate esigenze. Queste versioni non sono open source e vengono generalmente impiegate da grandi realtà aziendali. Sono altresì disponibili RDBMS open source, impiegati principalmente per lo sviluppo di applicazioni web data la loro semplicità d'uso e la vasta comunità di sviluppatori che se ne serve. Fanno parte di quest'ultima categoria alcune versioni più leggere destinate allo sviluppo mobile. Consentono di immagazzinare i dati localmente in uno o più file senza dover contattare un server remoto, sono impiegati soprattutto negli smartphone, PDA e in altri gadget elettronici.

Questa tipologia di database generalmente è garante delle proprietà ACID (Atomicità, Consistenza, Isolamento e Durabilità). Queste proprietà si applicano alle transazioni compiute dal database durante l'esecuzione delle query di creazione e modifica dei dati. Il principio di Atomicità impone che la transazione venga eseguita una volta sola e che, in caso di errore, non vengano alterati in alcun modo i dati. Per Consistenza si intende il rispetto e la coerenza dei vincoli di integrità, proprietà che deve essere rispettata prima e dopo che la transazione sui dati abbia avuto luogo. L'Isolamento comporta la possibilità di eseguire più transazioni in concorrenza senza generare incongruenze nel database. Le transazioni vengono quindi eseguite in modo indipendente e senza interferire le une con le altre. Il principio di Durabilità assicura che, al termine di una transazione eseguita con

successo, i dati vengano effettivamente memorizzati sul disco e non possano essere danneggiati da eventuali errori del software. I database relazionali sono basati su SQL, linguaggio estremamente versatile, ideale per query complesse. Inoltre, si basano su sistemi collaudati in quanto impiegati da diversi anni. Tuttavia, SQL rappresenta al tempo stesso anche uno svantaggio dato che necessita di uno schema fisso per poter operare al meglio. La struttura dati statica impone alcune limitazioni nella gestione di informazioni più articolate e complesse.

La categoria **NoSQL** impiega altre tipologie di strutture dati rispetto alla struttura tabellare utilizzata nei database relazionali. Queste basi di dati consentono di annidare le informazioni correlate all'interno di una singola struttura dati, inoltre permettono di archiviare enormi quantità di dati non strutturati, offrendo agli sviluppatori molta flessibilità. Con lo sviluppo del cloud computing, è sorta la necessità di distribuire i dati fra più server collocati in regioni differenti per rendere le applicazioni resilienti. Sono quindi stati sviluppati alcuni database NoSQL in grado di scalare orizzontalmente e di geo posizionare i loro dati con accortezza per venire in contro a queste esigenze.

I database di questa categoria vengono suddivisi in quattro principali tipologie, database di documenti, orientati alle colonne, di grafo e archivi chiave-valore. I database di documenti memorizzano le informazioni in documenti con struttura simile ad oggetti JSON (JavaScript Object Notation) [20]. Ogni documento contiene coppie di campi e valori assegnati. I valori possono essere di vari tipi inclusi elementi quali array e oggetti le cui strutture rimarcano gli oggetti impiegati dagli sviluppatori nel codice. Questa varietà di tipologie di strutture dati rendono i database estremamente flessibili, inoltre si adattano facilmente a modifiche alla struttura dati.

I database orientati alle colonne impiegano tabelle con colonne dinamiche per la persistenza dei dati. Nonostante l'impiego di tabelle, sono più flessibili dei database relazionali in quanto le righe hanno un numero di colonne dinamico. Il modello dati impiegato viene definito dalle query che verranno eseguite. La struttura dati è modellata attorno ad una o più query specifiche, per consentire di effettuare accessi in lettura il più rapidamente possibile.

Le basi di dati a grafo memorizzano le informazioni nei nodi e negli archi che li collegano. I nodi vengono impiegati per rappresentare le entità mentre gli archi descrivono le relazioni che connettono due nodi. Questa tipologia di database viene utilizzata per memorizzare informazioni in cui è necessario attraversare le relazioni per identificare particolari modelli nei dati.

Infine, i database a chiave-valore memorizzano i dati in oggetti contenenti una coppia chiave e valore. Questa tipologia di base di dati viene impiegata per l'immagazzinamento di grandi quantità di informazioni per le quali non è necessario effettuare query complesse in lettura. Le chiavi e i valori possono essere sia oggetti semplici che strutture dati più complesse.

I database NoSQL sono particolarmente indicati per l'utilizzo in sistemi distribuiti in quanto ottimizzati per scalare orizzontalmente. Durante lo sviluppo e l'impiego di un sistema distribuito per l'immagazzinamento di dati, si deve tenere in considerazione il teorema CAP (Consistency, Availability e Partition tolerance). Il teorema stabilisce l'impossibilità di garantire contemporaneamente tutte e tre le proprietà, perciò è impossibile che un sistema sia al tempo stesso consistente, sempre disponibile e tollerante ad eventuali guasti ai nodi di cui è composto. È possibile garantire al massimo solo due di queste caratteristiche. Questa categoria di database segue le proprietà BASE, ovvero i sistemi garantiscono la disponibilità (Basic Available) secondo il teorema CAP, ma il loro stato può variare nel tempo anche in assenza di input (Soft State) in quanto vengono effettuate operazioni interne al database durante un certo arco temporale per renderlo nuovamente consistente (Eventually consistent). I principali vantaggi di questa categoria sono la velocità e la possibilità di scalare orizzontalmente. La mancanza di uno schema fisso per la struttura dati è un vantaggio in quanto garantisce un'ampia libertà di azione per gli sviluppatori ma è al contempo anche uno svantaggio poiché non vengono garantiti i vincoli di integrità. Inoltre, questi sistemi non garantiscono di essere sempre consistenti, questo fattore può essere trascurabile in applicazioni quali i social network ma è fondamentale per applicativi bancari.

La persistenza dei dati nella piattaforma è affidata a tre database appartenenti alle categorie descritte in precedenza. La scelta di impiegare al tempo stesso sia database relazionali che NoSQL deriva dalla necessità di poter gestire al meglio le caratteristiche dei diversi tipi di dati trattati. Nello specifico, il sistema impiega attualmente un database relazionale, un database di documenti ed uno a grafo. **MySQL** [21], database management system di tipo relazionale (RDBMS), viene utilizzato per tenere traccia dei tessuti raccolti, dei risultati delle operazioni e degli esperimenti eseguiti sui campioni. MySQL è un software open source, disponibile da diversi anni quindi è ampiamente collaudato e supportato. Molti linguaggi di programmazione hanno librerie o funzioni dedicate per interagire col DBMS.

Ogni modulo del LAS dispone di un database relazionale ad esso associato nel quale, oltre a quanto descritto in precedenza, vengono memorizzate tutte le operazioni eseguite dagli utenti e viene effettuato il controllo dei dati modificati. Attraverso apposite tabelle di audit si tiene traccia di ciò che è stato eseguito, quando un determinato dato è stato

modificato e chi ha compiuto le modifiche al fine di accertare eventuali responsabilità o individuare utilizzi non corretti. Il database impiega una chiave univoca per ciascun record che consente di estrarre tutte le informazioni correlate ad una determinata entità mediante l'operazione di join nelle query. Inoltre, il sistema assegna un secondo identificativo univoco alle entità biologiche principali, la generazione del codice avviene in modo automatico da parte della piattaforma seguendo determinate regole. Questo codice viene utilizzato per connettere i dati relativi ad una determinata entità presenti nei database dei vari moduli in cui è stata trattata.

La scelta di impiegare MySQL è dettata dalla semplicità di configurazione e messa in funzione del database derivanti da anni di sviluppo. È ricco di documentazione sia online che cartacea e sono disponibili numerose applicazioni di terze parti, come ad esempio phpMyAdmin impiegata nel progetto LAS, per semplificare ulteriormente l'utilizzo del database. Inoltre, è una soluzione veloce e sicura. La velocità deriva dalle scelte implementative degli sviluppatori del database, i quali hanno optato per non inserire alcune funzioni di SQL a favore della velocità complessiva del sistema. Il database implementa la gestione degli utenti e dei permessi utente e dispone di appositi script per eliminare account anonimi ed eventuali database di test oltre a poter incrementare il livello di sicurezza della password dell'utente root.

Il secondo database impiegato è **Neo4J** [22], base di dati a grafo utilizzata principalmente per memorizzare i collegamenti fra i campioni di tessuti, i geni ed eventuali loro mutazioni. Neo4J è un software open source sviluppato in Java, può quindi essere impiegato su qualunque macchina supporti una Java Virtual Machine (JVM). I dati vengono interrogati attraverso chiamate REST indirizzate al processo che ospita il database. Le performance delle query sono strettamente legate alla struttura dati modellata. Perciò, la fase di progettazione è di fondamentale importanza, un buon design consente di massimizzare le prestazioni delle interrogazioni che verranno eseguite. Il grafo non ha uno schema predefinito, ogni nodo può contenere diverse proprietà ed essere contrassegnato da svariate etichette. I nodi che rappresentano entità della medesima tipologia saranno contraddistinti dalla stessa etichetta, questo consente di effettuare raggruppamenti sui dati. Le relazioni fra i nodi sono direzionali e sono anch'esse in grado di contenere proprietà mediante le quali effettuare specifiche query. Tutto ciò consente di memorizzare dati molto diversi fra loro in un'unica base dati. Nonostante la direzionalità, è possibile attraversare una relazione in entrambi i sensi mantenendo inalterate le prestazioni.

La regola fondamentale imposta da questa tipologia di database è l'assenza di collegamenti interrotti. Una relazione esistente non può puntare ad un nodo inesistente. La cancellazione di un nodo comporta quindi l'eliminazione di tutte le relazioni ad esso

associate. Il database della piattaforma risulta essere un insieme variegato di grafi. Ad esempio, come descritto in precedenza, l'operazione di derivazione porta alla generazione di una o più aliquote. Ogni aliquota presente nel sistema viene inserita in un grafo gerarchico, il nodo rappresentante l'aliquota di partenza viene quindi associato ai nodi nuovi attraverso il predicato "generates". Sono altresì presenti le relazioni fra gli istituti e i vari progetti di ricerca in essi condotti oltre alle relazioni fra gli utenti ed i vari gruppi di lavoro in cui sono coinvolti. Neo4j è il database a grafo open source con maggior compatibilità fra i vari linguaggi di programmazione backend, oltre a ciò offre supporto per le transazioni ACID. Questa tipologia di transazioni impone l'uso della regola precedentemente descritta, andando quindi a bloccare la cancellazione di nodi che hanno archi ad essi collegati.

Infine, la piattaforma fa uso di **MongoDB** [23], DBMS NoSQL orientato ai documenti. Il database è strutturato come insieme di documenti in stile JSON privi di uno schema dati fisso. Questo rende la struttura dati flessibile e di facile estensione per soddisfare nuove funzionalità. Qualora necessario, il database permette di imporre uno schema fisso ad un documento, al pari di una base dati relazionale, forzando determinati tipi di dato o rendendo alcuni campi obbligatori. I documenti all'interno di una collezione in MongoDB possono essere visti come una tabella in un database relazionale. La differenza sostanziale è la possibilità di avere documenti con strutture totalmente diverse rispetto alle tabelle la cui struttura (a righe e colonne) è fissa. Il database supporta le transazioni ACID su più documenti, questo permette ad un processo di effettuare modifiche bloccando eventuali altri processi. Inoltre, le modifiche ai documenti non sono visibili agli utenti finché tutte le operazioni sono concluse e viene effettuato commit o, in caso di errore, roll back della transazione. Consente di indicizzare qualunque campo per agevolarne le ricerche e può essere utilizzato altresì per l'implementazione di un file system.

È possibile configurare il sistema affinché gestisca in autonomia repliche dei dati presenti andando a verificare le copie in caso di errore o di un sovraccarico di lavoro. Questo consente di scalare orizzontalmente l'applicazione incrementandone l'affidabilità e la resistenza ad eventuali guasti. Le repliche dei dati hanno molteplici scopi, è possibile dedicare una replica alla gestione delle funzionalità dell'applicativo mentre una o più altre repliche vengono impiegate per backup, effettuare operazioni complesse o analisi sui dati.

In merito alla sicurezza dei dati, MongoDB offre funzionalità per la gestione dell'autenticazione e dell'autorizzazione degli utenti volte a limitare l'accesso ai soli dati di competenza. Dispone nativamente di un registro di controllo in cui tiene traccia delle attività e dei cambiamenti apportati. Oltre a ciò, permette di cifrare i dati memorizzati nel database così come le informazioni che vengono scambiate in rete. La piattaforma LAS

impiega attualmente questo database per la persistenza di metadati e configurazioni necessari all'utilizzo del modulo di query (MDDM). Nella categoria dei database a documenti, MongoDB si colloca fra quelli che sono supportati dalla maggioranza dei linguaggi di programmazione backend. L'integrazione con i sistemi in uso risulta quindi semplice e ampiamente documentata.

4 GenealogyID e studio delle entità

Per gestire correttamente tutti i dati e le procedure sperimentali, la piattaforma LAS identifica diversi tipi di entità e definisce un insieme predefinito di possibili relazioni fra loro. Ogni entità biologica è identificata da una chiave univoca e mnemonica, chiamata GenealogyID, che codifica le informazioni di rilievo riguardanti la storia dell'entità [24]. Il codice viene impiegato nei database relazionali associati a ciascun modulo per identificare l'entità con cui si sta operando. Questa chiave è composta da 26 caratteri alfanumerici e viene assegnata automaticamente dalla piattaforma seguendo determinati criteri per la sua generazione. Nello specifico, il codice è composto da dieci segmenti ciascuno dedicato ad una determinata caratteristica.

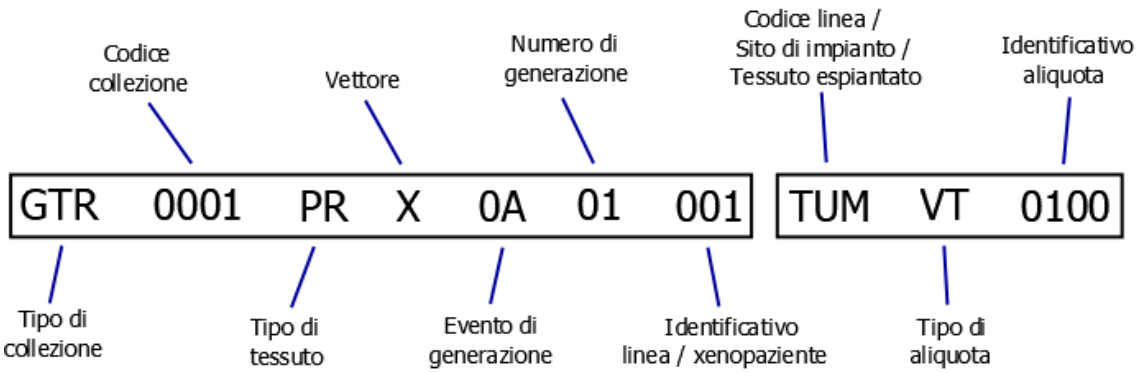

Figura 6: Struttura GenealogyID

In Figura 6 viene mostrato il codice identificativo di un'aliquota ottenuta mediante operazione di espianto condotta su xenopaziente. Nella prima parte del GenealogyID, composta da 17 caratteri, vengono riportate informazioni riguardanti i progenitori dell'entità, mentre nella seconda parte sono descritte alcune caratteristiche peculiari dell'entità corrente. Esaminando **la prima parte** del GenealogyID, si nota come i due segmenti iniziali contengano informazioni sulla collezione di origine quali il tipo di tumore ed un identificativo progressivo. Il database della piattaforma contiene un elenco che include la maggior parte dei tipi di tumore conosciuti. Per l'assegnazione dell'identificativo il sistema si basa su quanti campioni con lo stesso tipo di tumore in

oggetto sono già stati inseriti nel database e ne incrementa il valore. I segmenti successivi sono dedicati al tipo di tessuto tumorale ed al vettore di origine. Anche per questi due campi sono disponibili le tabelle contenenti i rispettivi elenchi. La tabella relativa al tipo di tessuto contiene entrambe le tipologie di tessuto, sia con metastasi che normale, così come alcuni elementi disponibili allo stato liquido ma considerati ugualmente tessuti (ad esempio il sangue). Per vettore si intende la provenienza del campione che può essere umana, animale o derivante dal protocollo impiegato per l'esecuzione di procedure su una linea cellulare.

Vanno a completare la prima parte del codice informazioni che contraddistinguono campioni derivanti da xenopazienti o linee cellulari. Questi dati sono organizzati in tre segmenti che tengono rispettivamente traccia dell'evento di generazione, del numero di generazioni successive e del codice della linea o dello xenopaziente. L'evento di generazione è un codice alfanumerico progressivo per il cui calcolo si tiene in considerazione il numero di volte in cui è stata utilizzata la collezione di partenza e lo stesso vettore. Il campo relativo al numero di generazioni successive tiene traccia del numero di volte in cui il campione è stato sottoposto ad operazione di impianto. Verrà quindi incrementato nel caso in cui venga effettuato l'impianto di un'aliquota di origine animale. L'ultimo segmento di questa parte del GenealogyID rappresenta un indice progressivo per tracciare quanti animali diversi sono stati trattati durante la stessa sessione. Qualora l'utente, durante la procedura di impianto, inserisca a sistema due o più cavie diverse, l'indice verrà incrementato per ciascun esemplare differente inserito.

La seconda parte del codice, composta dai restanti 9 caratteri, prevede alcuni segmenti impiegati per descrivere l'entità corrente. I segmenti presenti in questa seconda parte assumono quindi significati diversi a seconda dell'entità a cui il codice fa riferimento. Il primo segmento definisce la zona di impianto per gli xenopazienti, il tessuto di espianto per le aliquote generate con operazione chirurgica condotta sugli animali e, nel caso si tratti di linee cellulari, la configurazione impiegata. La zona di impianto ed il tessuto di espianto vengono selezionate dall'utente durante le relative procedure mediante apposito elenco precompilato dal sistema. La configurazione per le linee cellulari viene definita dall'algoritmo che calcola il GenealogyID per questo tipo di entità.

Gli ultimi due segmenti sono valorizzati esclusivamente per le aliquote, il primo campo sta ad indicare il tipo di aliquota, mentre il secondo rappresenta un indice progressivo. Nei codici che identificano xenopazienti o linee cellulari, questi ultimi due campi sono azzerati. Anche per il segmento relativo al tipo di aliquota è presente nel database una tabella in cui sono memorizzati tutti i valori che esso può assumere. L'indice progressivo viene incrementato qualora si inseriscano più campioni aventi le stesse caratteristiche ma

posizioni differenti sul piatto di conservazione. Per la determinazione di questo indice, il software prende in considerazione anche eventuali campioni aventi le stesse caratteristiche ma inseriti in precedenza mediante altre operazioni.

Questi segmenti vengono ulteriormente suddivisi qualora l'aliquota sia di tipo derivato, le suddivisioni vengono effettuate per tenere traccia del tipo di derivazione a cui è stata sottoposta l'aliquota. Nel caso di aliquote derivate il GenealogyID è quindi composto da dodici segmenti, mantenendo invariata la lunghezza complessiva. Questo accade in quanto i campioni di RNA ottenuti con la derivazione possono essere nuovamente sottoposti alla stessa operazione per ottenere aliquote di tipo complementary DNA o complementary RNA (cDNA, cRNA). Per le aliquote sottoposte per la prima volta a derivazione, vengono valorizzati solamente i due segmenti iniziali, il primo indicante il tipo di aliquota derivata ed il secondo contenente un indice progressivo. I restanti due campi vengono azzerati. Le aliquote derivate originate da altre aliquote di tipo derivato, vanno ad utilizzare tutti i campi a disposizione come descritto in precedenza, tenendo così traccia dell'aliquota di origine.

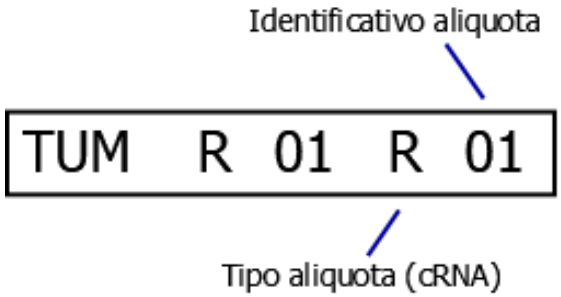

Figura 7: Dettaglio GenealogyID di un'aliquota derivata

La Figura 7 evidenzia la suddivisione operata sulla seconda parte del GenealogyID per differenziare le aliquote derivate. Il codice identificativo è strutturato in maniera tale per cui è possibile individuare velocemente a quale tipologia di entità fa riferimento. I campi presenti nella seconda parte del codice permettono agli utenti di determinare l'entità trattata. Nello specifico, qualora siano presenti i segmenti relativi al tipo di aliquota e all'identificativo correlato, l'entità trattata è un'aliquota. Nel caso in cui questi segmenti siano azzerati, si prende in considerazione il primo campo. Se il segmento contiene un codice relativo ad uno dei siti di impianto si tratterà di un biomouse, altrimenti il codice è riferito ad una linea cellulare.

Mediante comparazione della prima parte del GenealogyID è possibile individuare il rapporto gerarchico fra le entità. Questa gerarchia è visibile sottoforma di albero genealogico effettuando opportune query su Neo4j Browser, strumento disponibile con qualunque versione del database come supporto visivo per gli sviluppatori. La generazione viene inizializzata durante la procedura di impianto assegnando un valore ai segmenti relativi all'evento di generazione ed al numero di generazioni successive nel GenealogyID del biomouse. In particolare, il numero di generazioni successive viene impostato ad 1 mentre l'evento di generazione è associato a quanti biomouse differenti vengono creati in una stessa sessione. In seguito, ogni biomouse può dare origine a nuove aliquote che, qualora vengano impiantate in altri animali, andranno a generare biomouse di seconda generazione. Nel codice di queste entità rimane invariato l'evento di generazione ma si incrementa il numero di generazioni successive.

4.1 Entità e loro stati

In seguito all'analisi tecnico-funzionale della piattaforma si è proseguiti con lo studio delle entità in essa trattate. Ci si è concentrati su tre tipologie di entità e sui cambiamenti di stato che esse subiscono in seguito a determinate operazioni. Alcuni di questi cambiamenti sono delle vere e proprie mutazioni che portano alla creazione di nuove tipologie di campioni. Le entità analizzate sono le seguenti:

Aliquote, campioni raccolti e suddivisi in base alle loro caratteristiche intrinseche. Questi campioni vengono sottoposti ad operazioni differenti a seconda dello studio di ricerca e delle caratteristiche delle aliquote stesse. Come anticipato nei capitoli precedenti, i campioni possono quindi essere immagazzinati in appositi contenitori, sottoposti ad estrazione di aliquote derivate, impiantati in animali immunocompromessi o derivati in linee cellulari. Le aliquote possono essere campioni conservati allo stato solido, quali ad esempio frammenti di tessuto, oppure in stato liquido come una provetta di sangue. In entrambi i casi, è di fondamentale importanza la disponibilità del campione. Per i campioni a stato solido la disponibilità si traduce in un certo numero di frammenti mentre per i liquidi in un certo quantitativo di millilitri all'interno della provetta. Queste quantità verranno consumate durante l'esecuzione delle operazioni previste dal progetto di ricerca. Nel caso in cui il campione venga esaurito, verrà contrassegnato come non disponibile.

Tutto ciò si può tradurre in due stati primari caratterizzati dalla disponibilità o meno del campione all'interno della Biobanca.

Subordinatamente alla disponibilità del campione, si possono identificare diversi altri stati determinati dalla tipologia di aliquota trattata. I campioni sottoposti a procedura di derivazione danno origine ad aliquote di tipo derivato, questo stato è caratterizzato da valori di concentrazione e volume in quanto i campioni sono in forma liquida. Aliquote in stato derivato vengono impiegate per studi legati al DNA. Aliquote di tipo Formalin Fixed sono originate da tessuti conservati in formaldeide. Questa tipologia di conservazione permette di studiare i tessuti al microscopio. Per far sì che il campione resista per più tempo viene immerso in cera di paraffina generando aliquote di tipo Paraffin Section. I campioni in questo stato sono facilmente lavorabili per essere studiati al microscopio. Al termine dello studio vengono etichettati passando allo stato Labelled Section. I tipi di aliquota non elencati in precedenza vengono considerati campioni di tipo originale, queste aliquote sono impiegate come base per condurre le operazioni precedentemente descritte ed altri esperimenti.

Biomouse, complesso biologico generato dall'impianto di un'aliquota in un animale immunocompromesso. La piattaforma prevede l'utilizzo di topi in qualità di cavie per gli esperimenti. Qualora uno stesso animale sia sottoposto a più impianti in differenti parti del corpo, verranno generati biomouse differenti. Questa entità è strettamente correlata all'animale che ne dà origine, perciò analizzando gli stati dell'ospite si possono ricavare gli stati assunti dal complesso biologico. Nel momento in cui l'animale viene portato in laboratorio, l'utente incaricato procederà con l'assegnazione dello stato iniziale che può essere per allevamento o per la conduzione di esperimenti.

Gli animali destinati alla sperimentazione daranno origine ai biomouse a seguito dell'operazione di impianto di una o più aliquote. Il primo stato assunto dal complesso biologico è quindi lo stato etichettato come impiantato. Questo stato permette di effettuare trattamenti farmacologici sui tessuti impiantati, ma la loro esecuzione non incide sullo stato del biomouse che rimane invariato salvo il caso di morte accidentale del soggetto durante i test. In questo caso specifico, la piattaforma prevede l'assegnazione, da parte dell'utente, di uno stato apposito al biomouse. L'utente può assegnare anche altri stati in seguito al verificarsi di determinati eventi quali il trasferimento dell'animale in altra sede, la modifica della destinazione del soggetto da sperimentale ad allevamento e viceversa o la necessità di sopprimere l'animale per altri motivi. Al termine degli studi, l'utente può effettuare l'espianto di tessuti dallo xenopaziente. Quest'operazione può essere condotta in modo leggero, che non comporta la morte del soggetto e non necessita di preparazione

dello stesso, oppure in modalità normale. Sono quindi previsti i seguenti stati di espianto leggero, pronto per l'espianto ed espiantato per tracciare questa tipologia di operazioni.

Linee cellulari, esperimenti condotti in vitro. Queste entità possono venir generate da aliquote con opportune procedure di generazione o attraverso l'operazione di espansione di altre linee cellulari. Questa tipologia di entità non dispone di veri e propri stati in quanto viene impiegata per la conservazione e la replicazione di determinate cellule tumorali sulle quali verranno condotti appositi studi. È però possibile identificare alcuni eventi ai quali associare uno stato. Perciò, le operazioni di creazione delle linee cellulari determinano lo stato di disponibilità di queste nuove entità. Al contrario, l'operazione di eliminazione porta la linea cellulare nello stato non disponibile, in quanto l'entità viene fisicamente distrutta. La sperimentazione e l'archiviazione comportano una trasformazione della linea cellulare in una o più aliquote destinate ad altri studi, questa mutazione di entità determina la non disponibilità della linea cellulare impiegata. A seconda della procedura effettuata, vengono creati o utilizzati uno o più piatti di coltura contenenti le cellule. Perciò, una linea cellulare si considera non disponibile nel momento in cui vengono esauriti tutti i piatti da cui è composta.

La piattaforma gestisce al contempo altre entità quali i pazienti ed i consensi informati da essi rilasciati. Gestisce altresì i progetti di studio, entità a cui fanno riferimento i consensi informati e che stabiliscono le regole che gli utenti coinvolti dovranno seguire. I pazienti sono le persone che acconsentono alla raccolta dei campioni mediante operazione chirurgica. Per poter procedere con l'operazione è necessario che il paziente firmi il consenso informato. In assenza di questo documento è legalmente impossibile trattare i campioni o qualunque altra informazione legati al paziente. Fra le entità è inoltre possibile includere le collezioni di campioni. Esse rappresentano un insieme di campioni dalle caratteristiche omogenee o raccolti tutti in un unico evento. Un'operazione chirurgica condotta su un paziente in cui vengono raccolti differenti tipi di tessuti verrà catalogata in un'unica collezione. Queste entità sono state escluse da uno studio più approfondito in quanto non pertinenti ai fini del progetto di tesi.

4.2 Transizioni di stato

Le procedure eseguite sui campioni catalogati nella Biobanca danno origine a cambiamenti di stato che coinvolgono le aliquote impiegate. Alcune di queste operazioni comportano un vero e proprio passaggio da un'entità ad un'altra andando quindi a generare uno o più nuovi soggetti di studio. A seguito delle analisi precedentemente effettuate, è necessario disporre di una serie di aliquote per poter condurre un progetto di studio. Perciò, la prima operazione svolta su ogni campione si può definire come procedura di generazione dell'aliquota.

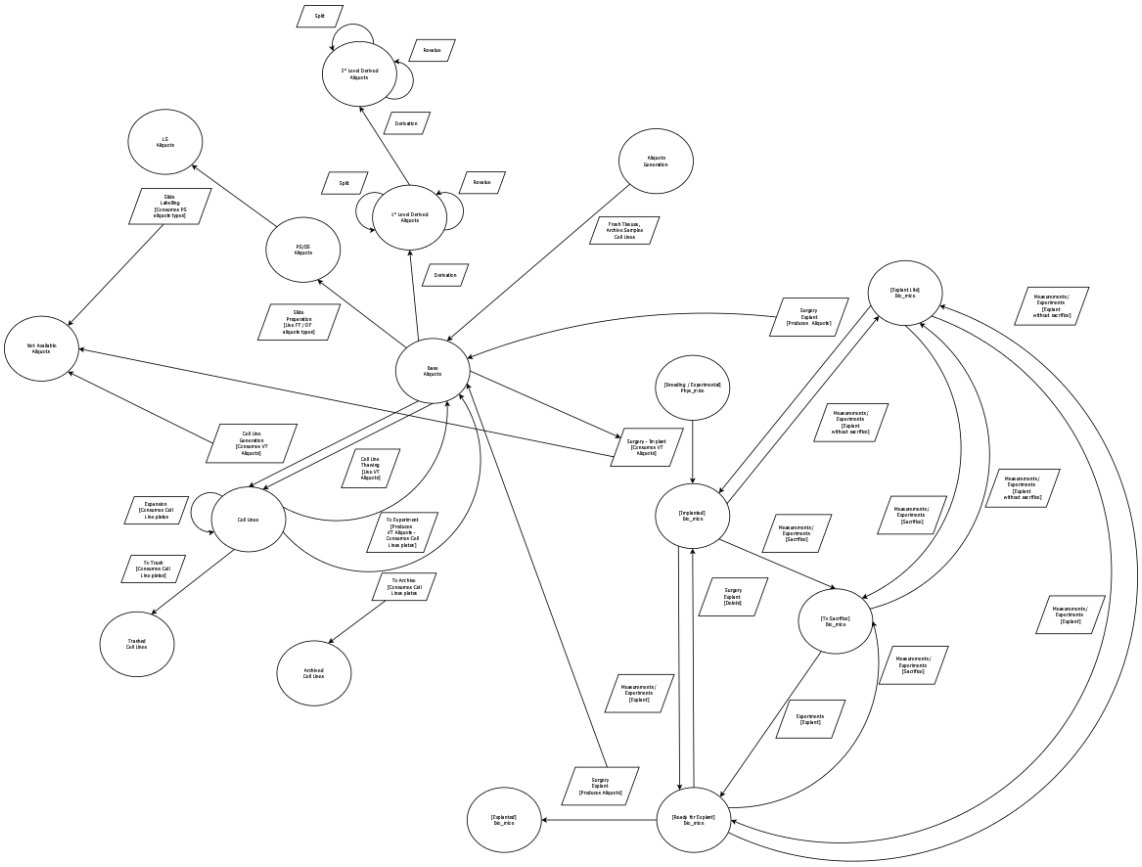

Figura 8: Schema di tutti i passaggi di stato in scala ridotta

Il sistema gestisce l'inserimento da fonti esterne di un'aliquota secondo le seguenti modalità: la raccolta di tessuti mediante operazione chirurgica e l'acquisizione di campioni provenienti da altri laboratori. L'operazione chirurgica può essere eseguita sia su persone che su animali, in questo secondo caso prende il nome di espianto. L'intervento chirurgico sugli umani non genera un vero e proprio cambiamento di stato ma è necessario affinché si possano collezionare campioni ed effettuare lo studio. La piattaforma consente inoltre la creazione di nuove aliquote partendo da campioni attualmente presenti nel sistema tramite l'archiviazione di una linea cellulare o altre procedure sui campioni.

Come precedentemente descritto, il sistema genera un identificativo univoco per ogni aliquota basandosi sulle caratteristiche di quest'ultima opportunamente inserite dall'utente. Qualora l'aliquota sia il risultato di un espianto condotto su xenopaziente o dell'archiviazione di una linea cellulare, il sistema terrà conto dell'identificativo dell'entità di origine durante la generazione del nuovo GenealogyID. In particolare, nel nuovo codice, verrà automaticamente riportata la prima parte dell'identificativo dell'entità di origine. Al contrario, la procedura di acquisizione dei campioni provenienti da altri laboratori prevede l'inserimento da parte dell'utente di tutte le informazioni relative all'entità di origine oltre alle caratteristiche correnti del campione. Il sistema procederà a calcolare il GenealogyID tenendo conto esclusivamente dei dati inseriti da parte dell'utente. Conclusa l'operazione di generazione, lo stato assunto implicitamente dall'aliquota può essere definito come disponibile, ovvero il campione è pronto per essere utilizzato nelle operazioni previste nel progetto di studio. Fra le operazioni condotte sulle aliquote, la **derivazione** è la procedura che permette di estrarre DNA, RNA o proteine da un campione. È inoltre possibile derivare ulteriormente un'aliquota di tipo RNA per ottenere complementary DNA o RNA (cDNA, cRNA).

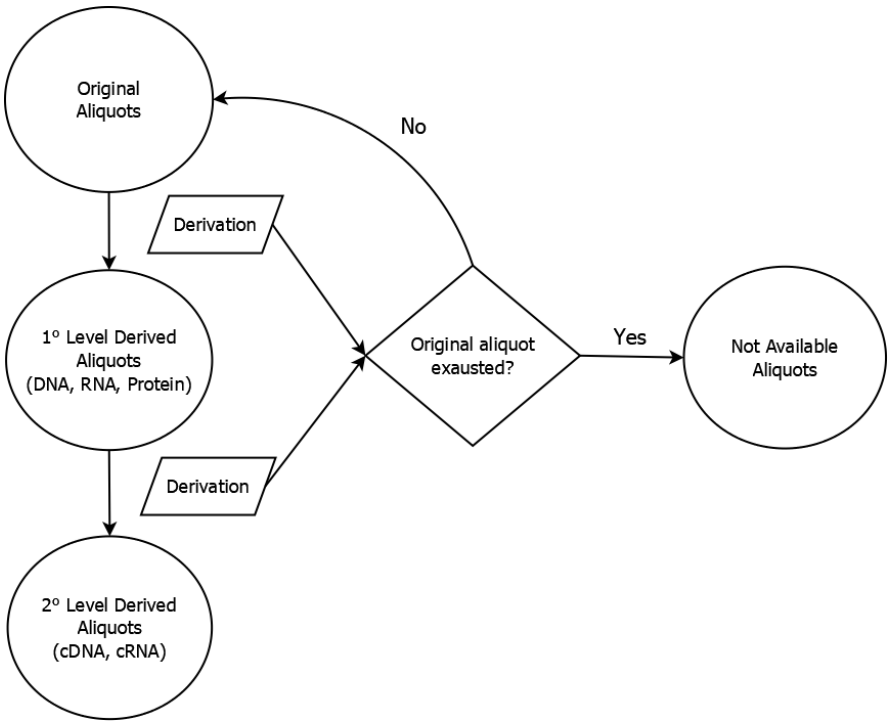

Figura 9: Diagramma derivazione

In Figura 9 è mostrato un estratto del diagramma a stati dell'intera piattaforma facente riferimento alla derivazione. Qualora l'operazione vada a consumare interamente l'aliquota di partenza, l'utente andrà a selezionare l'opzione aliquota esausta presente nella pagina, questo la rende non più disponibile nel sistema per future operazioni. Lo stato dell'aliquota originale viene quindi modificato in 'Non disponibile'. I nuovi campioni assumeranno lo stato di aliquote derivate disponibili e verranno utilizzate per condurre esperimenti. Dal punto di vista del GenealogyID, le nuove aliquote condividono buona parte del codice del campione di origine, viene effettuato l'aggiornamento della seconda parte del codice. In particolare, vengono modificati il tipo di aliquota e l'indice progressivo in corrispondenza del tipo di derivazione eseguita. Si può quindi considerare la derivazione altresì come un'operazione di generazione in quanto vengono prodotte una o più aliquote derivate che vanno ad aggiungersi alle altre collezioni conservate nella Biobanca.

Altra operazione che consente la creazione di nuove aliquote è lo **split**. Come suggerisce il nome, questa procedura consente di creare uno o più campioni andando a dividere un'aliquota, è eseguibile soltanto su aliquote di tipo derivato in quanto sono campioni allo stato liquido. L'utente seleziona quante nuove aliquote intende creare e per ognuna, può

specificare la concentrazione ed il volume tenendo in considerazione i rispettivi valori dell'aliquota di partenza. Anche per questa operazione, qualora l'aliquota di origine venga esaurita, l'utente andrà a selezionare l'opzione aliquota esausta ed il sistema la renderà non più disponibile. La procedura genera quindi una o più aliquote aventi lo stesso tipo del campione originale. Il codice identificativo di questi nuovi campioni è pressoché identico a quello dell'aliquota di origine, l'unica modifica apportata è la generazione di un nuovo indice progressivo presente nella seconda parte del GenealogyID in corrispondenza della derivazione effettuata sul campione di partenza.

Fra le operazioni di generazione è possibile includere la slides preparation. Questa procedura è utilizzata nella preparazione dei vetrini di laboratorio per studi condotti al microscopio. Fa uso di aliquote di tipo "Formalin Fixed" o "OCTFrozen" per generare vetrini rispettivamente di tipo "Paraffin Section" e "OCTSection". Come visto in precedenza, qualora il campione di origine venga esaurito, l'utente andrà ad informare il sistema tramite apposito flag della non disponibilità dell'aliquota. Lo slides labelling è, al contrario, una procedura di trasformazione delle aliquote nella quale l'utente va ad etichettare i vetrini di tipo "Paraffin Section" generando aliquote di tipo "Labelled Section". Questa operazione va a sostituire del tutto l'aliquota di partenza rendendola non più disponibile per altre operazioni. Anche in questo caso, il nuovo codice identificativo creato differisce solamente per la seconda parte del GenealogyID, nello specifico il tipo di aliquota.

Le procedure descritte in precedenza operano tutte sulle aliquote e possono portare ad un cambiamento di stato da disponibile a non disponibile qualora il campione utilizzato come base venga esaurito. Le aliquote create vengono inserite nell'albero genealogico dei campioni memorizzato su Neo4j in qualità di nuove foglie. Lo stato non disponibile si ripercuote sul contenitore dell'aliquota, generalmente sul tubetto che la contiene, rendendo quest'ultimo a sua volta non disponibile per l'archiviazione di altri campioni.

La procedura di trasferimento dei campioni è stata esclusa dallo studio in quanto non genera cambiamenti di stato ma comporta un cambiamento di proprietà del campione. L'utente che trasferisce un'aliquota andrà a selezionare un utente destinatario fra quelli disponibili nel database della piattaforma, verranno quindi aggiornate le informazioni relative al proprietario ed eventualmente, al contenitore dell'aliquota.

La rivalutazione delle aliquote derivate è anch'essa un'operazione esclusa dallo studio in quanto non ha come risultante un cambiamento di stato per i campioni coinvolti. La procedura ha come scopo il perfezionamento dei parametri di concentrazione e volume

caratterizzanti questa tipologia di aliquote e misurati durante la derivazione. Quindi, questo aggiornamento non influisce sullo stato del campione.

La procedura di **impianto** di un tessuto tumorale in una cavia da laboratorio porta ad un passaggio di entità piuttosto che ad un cambiamento di stato. In questa operazione vengono utilizzati un'aliquota conservata in vitale ed un animale immunocompromesso per generare una nuova entità, il biomouse. Al fine di poter effettuare questa e le seguenti procedure, è necessario aver registrato gli animali nel sistema. Durante la fase di registrazione dei topi nella piattaforma, l'utente andrà a scegliere la loro destinazione che può essere per allevamento (*breeding*) o per effettuare esperimenti (*experimental*). In questa fase preliminare vengono inseriti anche altri parametri quali il sesso e l'età dell'animale oltre al codice identificativo ad esso riservato. Questo codice non è legato al GenealogyID derivante da impianto ma viene utilizzato esclusivamente per identificare l'animale all'interno del modulo.

Al termine della procedura di impianto, il sistema aggiorna lo stato dell'animale, memorizzando il passaggio da *experimental* ad *implanted*. Inoltre, la piattaforma assegna automaticamente un identificativo per il biomouse appena creato, partendo dal GenealogyID dell'aliquota utilizzata nell'impianto. Nel nuovo codice vengono mantenuti i primi segmenti riguardanti la collezione ed il tipo di tessuto di origine mentre viene azzerata totalmente la seconda parte in quanto l'entità trattata non è un'aliquota. Il sistema imposta quindi il segmento riservato al vettore inserendo la sigla dedicata agli animali.

I parametri successivi vengono calcolati utilizzando come base i rispettivi segmenti, qualora presenti, nell'id dell'aliquota impiantata. Nello specifico, qualora l'aliquota impiantata sia di origine umana, tutti i segmenti vengono inizializzati. Il parametro relativo all'evento di generazione viene successivamente incrementato qualora nella Biobanca siano presenti uno o più biomouse originati dalla stessa collezione dell'aliquota utilizzata per l'impianto. Nel caso in cui l'aliquota impiantata sia di origine animale, il sistema mantiene invariato il codice che identifica l'evento di generazione ma incrementa il numero di generazioni successive. Infine, per le aliquote originate dall'archiviazione di una linea cellulare, il sistema mantiene invariato il codice dell'evento di generazione ed inizializza il numero di generazioni successive. In tutti i precedenti casi, l'ultimo segmento che identifica lo xenopaziente viene incrementato qualora vengano registrati in una stessa sessione più impianti effettuati su animali diversi.

La procedura di **espianto**, al contrario dell'operazione di impianto, utilizza come base di partenza un biomouse per ottenere una o più aliquote. In questo caso si ha quindi il passaggio di entità inverso rispetto all'operazione precedente. Come la maggior parte

delle procedure presenti nel sistema, per poter essere eseguita necessita di una fase iniziale di preparazione. In questa fase viene preparato lo xenopaziente all'operazione, viene modificato lo stato che passa da *implanted* a *ready for explant*.

La procedura di espianto genera come risultato aliquote di vari tipi, le quali avranno in comune la parte iniziale del GenealogyID derivante dal codice del biomouse impiegato. Viene così tenuta traccia della collezione di origine del biomouse, mentre nella seconda parte del codice è riportato il tipo di tessuto asportato dall'animale. Infine, il segmento contenente il vettore riporterà la sigla dedicata agli xenopazienti. L'operazione va inoltre a modificare lo stato dell'animale ospite effettuando il passaggio dallo stato *ready for explant* allo stato *explanted*. A conclusione dell'operazione chirurgica, il sistema aggiorna i dati dell'animale inserendo come data del decesso la stessa dell'espianto. Questo avviene in quanto l'espianto conduce alla morte della cavia impiegata, il sistema di conseguenza, rende l'animale non più disponibile per nuove misurazioni.

Durante la pianificazione della procedura di espianto, il ricercatore può scegliere di eseguire l'operazione in forma leggera, ovvero senza causare la morte del soggetto. In questa prima fase la piattaforma va a modificare lo stato dell'animale che passa quindi da *implanted* ad *explantLite*. Questa operazione, al pari dell'espianto classico, consiste nel prelievo di tessuto tumorale dall'animale e la conseguente creazione di nuove aliquote. Dal punto di vista del software la procedura è quasi del tutto identica all'espianto classico. Entrambe le procedure fanno uso della stessa interfaccia grafica, la funzione di memorizzazione dei dati aggiorna di conseguenza la data di decesso dell'animale basandosi sul tipo di espianto eseguito. Il prelievo vero e proprio, eseguito dal ricercatore in laboratorio, viene tuttavia effettuato con tecniche non invasive al fine di garantire la sopravvivenza dell'animale. Al termine, lo stato del xenopaziente viene riportato ad *implanted* ed è quindi possibile effettuare nuove misurazioni e trattamenti.

Gli animali in laboratorio possono assumere altri stati oltre a quelli descritti in precedenza, questo indipendentemente dall'essere stati sottoposti ad operazione di impianto. Questi stati dipendono dal verificarsi di condizioni esterne al software e quindi devono essere opportunamente inseriti dall'utente. Il trasferimento delle cavie in altre strutture prevede lo stato *transferred*. Mentre in caso di morte accidentale viene assegnato lo stato *dead accidentally*. Qualora sia necessario sopprimere il soggetto per motivi diversi dall'espianto viene utilizzato lo stato di *sacrified*. Gli utenti possono inoltre assegnare lo stato da sacrificare (*toSacrifice*) per contrassegnare l'animale in modo che non venga utilizzato per altre operazioni. Queste modifiche non incidono sul GenealogyID di eventuali biomouse impiantati, ma in caso di morte accidentale, non sarà possibile effettuare l'espianto, sia in forma leggera che normale, precedentemente

programmato. Data la natura e le conseguenze dell'operazione, non è possibile modificare lo stato di un animale dopo aver eseguito l'espianto.

Nella piattaforma viene gestito un ulteriore cambiamento di entità ovvero il passaggio da aliquota a linea cellulare e viceversa. La creazione di una **linea cellulare** avviene con le operazioni di generazione e thawing effettuate utilizzando come base di partenza un'aliquota. Entrambe le procedure impiegano campioni conservati in vitale, più precisamente con la procedura di thawing viene generata una linea cellulare partendo da un'aliquota ottenuta in precedenza con processo di archiviazione. L'aliquota impiegata nell'operazione di thawing può quindi aver origine solamente dall'archiviazione di un'altra coltura di cellule. Al contrario, il processo di generazione di una coltura cellulare fa uso esclusivamente di un'aliquota di origine umana oppure di origine animale, a condizione che sia conservata in vitale. Il campione viene consumato nella sua interezza, al termine della procedura è posto nello stato di non disponibile.

La creazione del GenealogyID per le nuove linee cellulari segue quanto descritto in precedenza per i biomouse. Non trattandosi di aliquote, i segmenti presenti nella seconda parte del codice vengono azzerati, fatta eccezione per il segmento relativo al codice della coltura. Nel caso del thawing, il nuovo GenealogyID riprenderà buona parte del codice dell'aliquota di origine ed andrà ad incrementare il numero di generazioni successive, mantenendo invariato il segmento relativo all'evento di generazione. Qualora l'aliquota di origine della coltura sia derivante da uno xenopaziente, e quindi siano presenti i campi relativi all'evento di generazione ed al numero di generazioni successive, l'algoritmo di generazione del GenealogyID inizializzerà nuovamente il numero di generazioni successive. Per l'assegnazione del codice relativo all'evento di generazione, terrà conto di eventuali colture appartenenti alla stessa collezione ed incrementerà opportunamente il valore. Si ha quindi un reset dei segmenti relativi alle generazioni precedenti dovuto al cambiamento di entità trattata che porta quindi alla generazione di un nuovo lignaggio.

Entrambe le procedure hanno come risultato la creazione di uno o più piatti di coltura delle cellule impostando di fatto il loro stato a disponibile. Queste piastre di coltura sono le unità base da cui è composta una linea cellulare, per ogni futura operazione vengono impiegati uno o più piatti interi, pertanto non sono divisibili. La generazione ed il thawing sono subordinati alla creazione di protocolli per la gestione delle condizioni delle colture. Questi protocolli determinano l'ambiente di coltura delle cellule andando a definirne i nutrimenti, gli ormoni, gli antibiotici ed altre sostanze da includere nella base delle piastre per supportare lo sviluppo delle cellule. Viene inoltre definito il tipo di processo utilizzato per la generazione che verrà riportato nel GenealogyID della linea cellulare nel segmento relativo al vettore.

La procedura di **espansione** è un'operazione che porta alla generazione di nuove linee cellulari partendo da una coltura esistente. L'utente dovrà selezionare un protocollo per la definizione dell'ambiente delle nuove piastre e andrà ad impostare un fattore di diluizione per la coltura di partenza selezionata. I protocolli impiegati per l'espansione sono i medesimi utilizzati nel thawing, questo perché le procedure hanno come obiettivo la creazione di nuove colture partendo da una base derivante direttamente da un'altra linea cellulare. Il fattore di diluizione rappresenta il rapporto fra le piastre risultanti ed i piatti impiegati per condurre la procedura. Pertanto, verranno create m nuove piastre andando a consumarne n della linea cellulare esistente. L'operazione comporta un cambiamento di stato per la coltura di partenza qualora vengano esauriti tutti i piatti, in questo caso la linea cellulare non sarà più disponibile. Il GenealogyID della nuova coltura viene aggiornato inserendo il nuovo vettore, qualora il protocollo di espansione utilizzi un processo di generazione diverso. Viene altresì aggiornato il segmento relativo all'identificativo di linea cellulare, nello specifico risulta essere il valore dello stesso segmento della coltura di origine incrementato di uno.

La piattaforma permette di impiegare contemporaneamente due o più protocolli diversi per effettuare l'espansione della coltura. In questo caso, il valore m (nuove piastre) del fattore di diluizione è dato dalla somma dei piatti destinati a ciascuna nuova linea cellulare. Per condurre nuove sperimentazioni sulle colture cellulari, il sistema mette a disposizione una funzione che effettua automaticamente il passaggio da linea cellulare ad aliquota. Questa funzionalità impiega un certo numero di piastre della coltura scelta per creare lo stesso numero di aliquote conservate in vitale ed utilizzabili come base per altri esperimenti. Per ogni piastra utilizzata viene quindi creata una nuova aliquota. Come per le operazioni precedenti, qualora il numero di piastre impiegate risulti pari alla massima quantità presente nella Biobanca, la coltura risulterà non più disponibile. Il GenealogyID dei nuovi campioni conterrà il segmento del tipo aliquota valorizzato al tipo di conservazione in vitale (VT) ed il segmento relativo all'indice progressivo incrementato per ogni nuovo campione. La prima parte del codice rimarca interamente l'identificativo della linea cellulare di origine.

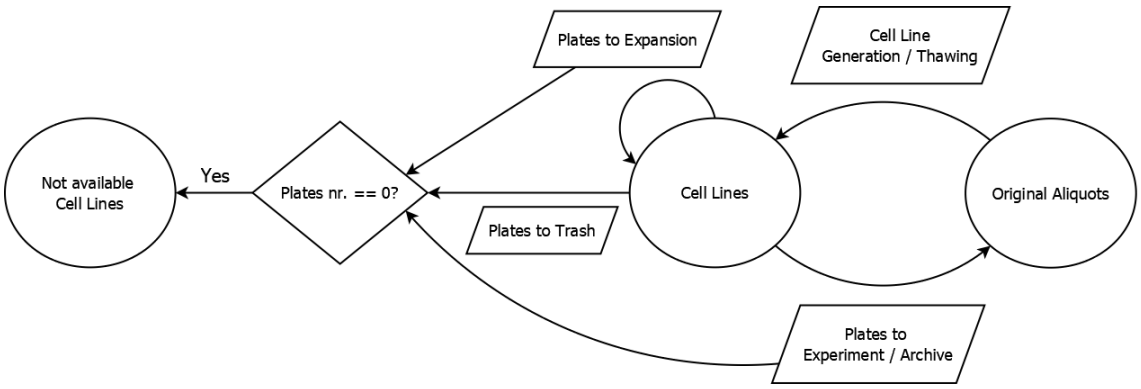

Figura 10: Stati delle colture cellulari

Come mostrato in Figura 10, l'attuale versione del software permette di effettuare due ulteriori operazioni sulle linee cellulari, l'eliminazione e **l'archiviazione**. Con l'eliminazione l'utente indica un certo numero di piastre che vengono rimosse dalla Biobanca e successivamente distrutte. Mentre l'archiviazione è l'operazione inversa alla generazione e permette di creare nuove aliquote utilizzando uno o più piatti di una linea cellulare. Al pari della sperimentazione, il GenealogyID dei nuovi campioni riprenderà interamente la prima parte del codice della coltura di origine ed andrà ad integrare l'indice progressivo ed il tipo di aliquota a seconda del metodo di conservazione scelto.

Attraverso questa procedura è possibile creare tutte le tipologie di aliquote gestite dalla piattaforma ad eccezione dei campioni di tipo derivato. Come analizzato in precedenza, per ottenere questa tipologia di aliquote è necessario schedulare l'operazione di derivazione ed impiegare le varietà di campioni e gli strumenti previsti dal protocollo scelto. Attualmente non sono previsti protocolli di derivazione operanti direttamente sulle linee cellulari, pertanto il sistema non permette la creazione di questa tipologia di campioni mediante l'archiviazione. In Figura 10 viene evidenziato come il principale cambiamento di stato subito dalle linee cellulari sia determinato dal numero di piastre disponibili nella Biobanca. Ogni operazione effettuata comporta quindi l'utilizzo o la creazione di un certo numero di piatti della coltura che va a modificarne la disponibilità. Il sistema tiene traccia di questo indice, qualora il valore raggiunga lo zero, la linea cellulare verrà considerata esaurita e non più disponibile.

4.3 Diagramma degli stati e considerazioni

Le transizioni di stato riscontrate nella piattaforma nonché i passaggi fra le entità oggetto di studio sono state rappresentate in un diagramma a stati. Lo scopo principale di questo schema è l'individuazione delle principali operazioni che portano ad una modifica dello stato di un campione o che causano un cambiamento di entità. Per la realizzazione di questo diagramma si è proceduto con lo studio del funzionamento della piattaforma e del codice impiegato per eseguire le operazioni. L'analisi ha riguardato anche le variazioni effettuate ai dati presenti nei vari database in seguito ad ogni step condotto sulla piattaforma.

Lo studio delle modifiche apportate alle informazioni memorizzate ha permesso di individuare i parametri principali che determinano gli stati nonché le varie tipologie dei campioni. I passaggi di stato sono stati individuati tramite l'analisi congiunta delle tabelle interessate nelle operazioni e le relative tabelle di audit. Queste ultime contengono generalmente l'ultimo stato assunto dal campione prima di essere sottoposto alla procedura. Le tabelle di audit permettono quindi di individuare le operazioni che modificano i parametri principali rispetto a quelle che effettuano un aggiornamento dei dati piuttosto che una modifica ad informazioni di importanza secondaria limitatamente al progetto di tesi. La struttura della piattaforma consente di individuare facilmente i passaggi fra le entità in quanto impiega moduli diversi per la gestione delle colture e dei biomouse.

Ciascun modulo dispone del proprio database relazionale in cui memorizza i dati trattati. L'elemento di congiunzione fra i vari database è rappresentato dalle aliquote identificate mediante il loro GenealogyID. Entrambi i moduli impiegano una tabella di supporto in cui tengono traccia dei campioni che danno origine alle rispettive entità. Allo stesso modo, ogni aliquota generata mediante archiviazione o espianto verrà memorizzata sia nel modulo in cui è prodotta sia nel database della Biobanca. Questa ridondanza di informazioni è mitigata dalla caratteristica di univocità del codice identificativo oltre a semplificare alcune procedure in cui è necessario elaborare il GenealogyID dell'aliquota di origine. Lo studio diretto delle funzioni impiegate dall'applicativo ha permesso di capire nel dettaglio come vengono eseguite le procedure oltre a chiarire alcuni aspetti legati all'assegnazione dei valori ai segmenti presenti nel GenealogyID delle entità. Inoltre, l'analisi del software per la realizzazione del diagramma ha portato all'identificazione dei dati elaborati da ogni singolo step delle operazioni che prevedono più fasi.

Queste informazioni sono state impiegate anche per la realizzazione della dashboard. Le Figure 9 e 10, riportanti gli schemi della derivazione e delle colture cellulari, sono un estratto del **diagramma a stati** generale riportato in Figura 8. Lo schema sottolinea la centralità delle aliquote e la loro importanza all'interno di una Biobanca. Nei paragrafi precedenti, si può notare come i passaggi fra le entità implicano tutti l'utilizzo o la generazione di una o più aliquote, la creazione di un biomouse non può quindi avvenire impiegando direttamente le cellule contenute in una coltura. Pertanto, vengono trattati esclusivamente passaggi di entità da e verso le aliquote.

Le trasformazioni subite dai campioni hanno come principale scopo la generazione di diverse tipologie di aliquote per la conduzione di specifici esperimenti. Ad esempio, le aliquote derivate sono impiegate per lo studio del genoma del tumore e di eventuali variazioni subite a seguito dei trattamenti somministrati, mentre i campioni all'interno dei vetrini vengono studiati al microscopio per individuare trasformazioni morfologiche delle cellule tumorali. La sperimentazione in vitro, ovvero mediante linee cellulari, permette di studiare il modo in cui le cellule si moltiplicano in un ambiente controllato. Assume quindi un'importanza fondamentale la definizione dei parametri per i protocolli di generazione ed espansione impiegati nella creazione delle colture. L'impiego di protocolli differenti per la generazione di linee cellulari basate sulla stessa tipologia di campioni, permette di analizzare le differenze fra gli sviluppi di ciascuna coltura. Questo consente di individuare sostanze che inibiscono la crescita delle cellule cancerogene.

Nel diagramma viene altresì evidenziato come il biomouse sia un complesso biologico generato dall'impianto di un campione di tessuto tumorale in un animale immunocompromesso. La nuova entità generata è quindi strettamente correlata al ciclo vitale dello xenopaziente da cui trae origine. Inoltre, è alla base di tutte le operazioni che verranno eseguite sull'animale. La sperimentazione in vivo, ovvero l'impiego dei biomouse, permette di studiare la risposta dei tessuti tumorali in seguito alla somministrazione di farmaci. I ricercatori, attraverso le procedure di impianto ed espianto, possono studiare le differenze fra i campioni impiegati prima dell'impianto e le aliquote ottenute mediante espianto. Lo studio di queste ultime porterà a stabilire se il risultato del trattamento condotto sul soggetto di ricerca è stato efficace o meno.

Effettuando un'analisi macroscopica dello schema si può individuare come la gestione dei campioni avvenga in modo circolare fra le entità prese in considerazione. Ciascuna collezione può essere studiata contemporaneamente attraverso la generazione di colture cellulari e l'impianto su cavie. Le aliquote derivanti dall'archiviazione e dall'espianto delle rispettive entità possono essere impiegate nuovamente per lo stesso ciclo oppure dare origine ad una nuova serie che coinvolge un'entità diversa dalla precedente. Questi

passaggi, e le gerarchie che ne conseguono, sono visibili materialmente attraverso determinate query su Neo4j Browser. Il confronto dei risultati ottenuti in seguito alle procedure svolte può portare all'individuazione di nuove tipologie di cure o alla scoperta di nuovi punti deboli delle cellule tumorali.

5 Dashboard

Il progetto della dashboard nasce dalla necessità di avere un modulo di sintesi delle attività che vada ad estendere l'attuale implementazione del LAS. A seguito di indagini condotte sugli utenti del sistema, si è notata una costante richiesta di un'interfaccia che permetta una visualizzazione più immediata delle informazioni distribuite fra i vari moduli. Inoltre, questa funzionalità aggiuntiva era stata prevista dagli sviluppatori della piattaforma in una seconda versione, per integrare ed ampliare le funzionalità offerte dal software oltre a renderlo più facilmente utilizzabile. Lo scopo principale del nuovo modulo consiste nel mostrare all'utente una pagina in cui può visualizzare rapidamente il resoconto delle attività in corso. Inoltre, l'utente deve essere messo in condizione di poter riprendere ad operare su uno qualsiasi dei task mostrati. Oltre alle procedure in sospeso, è richiesta la visualizzazione di alcune statistiche integrative sulle collezioni presenti nell'intero database.

Visto il gran quantitativo di collezioni raccolte in un sistema di produzione, si è deciso di ridurre al minimo le informazioni accessorie al fine di rendere il reperimento delle statistiche il più rapido possibile. L'analisi funzionale della piattaforma ha permesso di individuare le operazioni che sono suddivise in più fasi e in quali step viene permesso all'utente di interrompere il lavoro. Questo nuovo componente deve quindi interfacciarsi con i moduli della Biobanca e degli Xenopazienti per raccogliere informazioni riguardanti il numero di attività in sospeso e lo stato in cui ciascuna di esse si trova. Oltre a ciò, viene consultato il MDDM (MultiDimensional Data Manager) per il reperimento delle statistiche sui campioni raccolti, questo perché il modulo è ottimizzato per l'esecuzione di query che elaborano grandi quantità di dati.

Le principali attività mostrate agli utenti riguardano le procedure sulle aliquote e lo stato delle cavie. Per questa categoria di informazioni la piattaforma adotta due tipologie diverse di accessi e visibilità all'interno dei moduli che sono state riprese nel modulo di sintesi. In particolare, le operazioni sulle aliquote vengono assegnate ad uno specifico utente durante la fase di pianificazione e quest'ultimo le dovrà portare a termine per conto proprio, queste procedure non sono pertanto visibili agli altri utenti. Gli xenopazienti sono disponibili per tutti gli utenti che hanno accesso al laboratorio fintanto che non sono stati sottoposti alla procedura di impianto. Nel momento in cui l'utente di un determinato gruppo sottopone ad operazione di impianto una cavia, l'animale ed il biomouse prodotto

diventano di proprietà del gruppo a cui l'utente appartiene. Quindi, tutti gli utenti appartenenti a quel determinato gruppo avranno visibilità e potranno effettuare le misurazioni o le procedure previste dal protocollo sul soggetto trattato. L'accesso al modulo di riepilogo è pertanto consentito esclusivamente agli utenti che possono operare nella Biobanca, ovvero gli utenti base e i PI (Pricipal Investigator). Le informazioni legate alle statistiche delle collezioni sono a carattere generale, perciò non è stato inserito nessun controllo per quanto riguarda la visualizzazione di collezioni appartenenti a gruppi di lavoro differenti.

5.1 Progettazione

Nella fase di progettazione si è presa in considerazione la tipologia di dati trattati, su questa base è stata valutata una possibile interfaccia grafica per rappresentarli al meglio. A seguito di queste osservazioni si è proceduto con l'analisi della struttura del backend e la scelta delle tecnologie per il conseguente sviluppo. Lo studio sulle informazioni che verranno mostrate agli utenti ha evidenziato due categorie di dati, in particolare sono presenti informazioni che possono assumere diversi stati e una seconda tipologia di dati a carattere più generale e priva di stato. Nella prima categoria rientrano le procedure in sospeso riguardanti le aliquote e gli stati assunti dagli Xenopazienti, mentre le statistiche riguardanti le collezioni fanno parte della seconda tipologia di dati. Le informazioni appartenenti alla prima categoria possono essere sintetizzate in una serie di contatori che tengono traccia di quante operazioni o biomouse sono in un determinato stato.

In merito alle statistiche sulle collezioni, si è scelto di mostrare quante raccolte diverse di ciascun tipo di tumore sono presenti mediante l'uso di un diagramma. Lo studio del modulo della Biobanca ha evidenziato sei operazioni principali, quali derivazione, split, la preparazione e la marcatura dei vetrini, la rivalutazione ed il trasferimento delle aliquote. Mentre per il modulo degli Xenopazienti sono stati individuati quattro casi di interesse per gli utenti, ovvero le cavie disponibili per l'impianto, sottoposte ad impianto, gli animali sotto trattamento e con espianto programmato. Sulla base di queste informazioni si è proceduto con la definizione di un'interfaccia grafica che racchiuda in una singola schermata tutti i dati precedentemente elencati. La pagina, oltre a mostrare queste informazioni, deve permettere all'utente di poter riprendere il lavoro laddove è stato interrotto. Per soddisfare questo requisito, si è deciso di suddividere la pagina in due

aree dedicate rispettivamente al controllo ed alla visualizzazione dei dati. Per la definizione della struttura di entrambe le parti della pagina si è adottato un approccio orientato alla facilità d'uso, con informazioni chiare ed essenziali affiancate da pulsanti per il controllo delle funzioni messe a disposizione dalla dashboard. Il funzionamento ipotizzato per il modulo di riepilogo è il seguente: l'area di controllo deve mostrare una serie di pulsanti, ciascuno di essi va ad attivare una specifica funzione per la compilazione dell'area di visualizzazione con i dati richiesti. In questa seconda parte, l'utente deve poter visualizzare un breve riepilogo e riprendere l'attività in sospeso mediante appositi pulsanti.

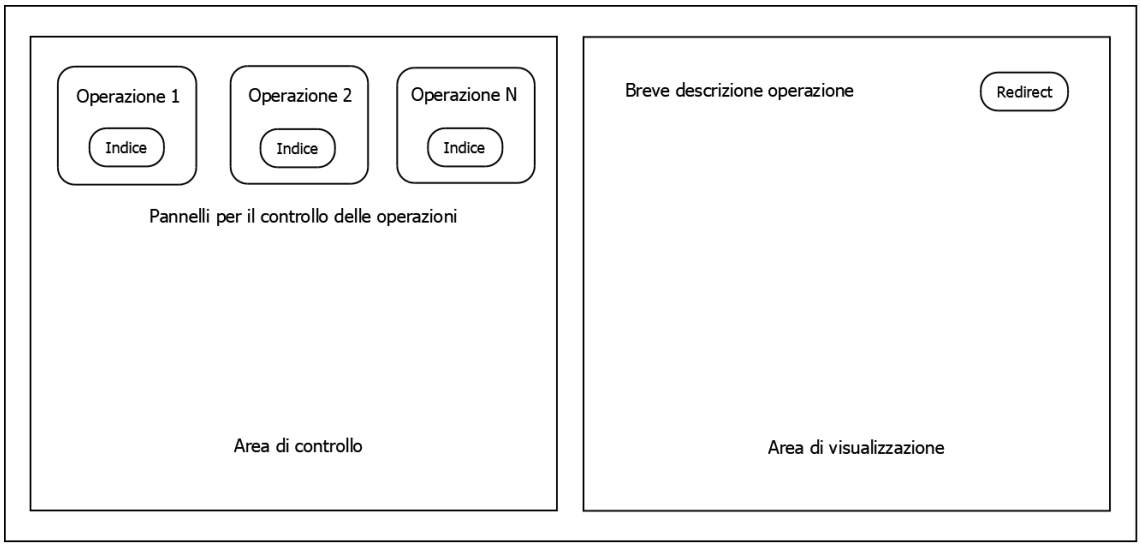

Figura 11: Struttura prevista inizialmente per la dashboard

Per dare continuità all'applicativo, si è deciso che la struttura della pagina più in generale debba seguire quanto fatto nel resto del software, mantenendo quindi gli stili della navbar e del footer. Mentre il body della pagina è stato modificato, rispetto al resto della piattaforma, per occupare al meglio lo spazio in verticale. In base al funzionamento ipotizzato per la dashboard, si è proceduto alla definizione della struttura dell'area di controllo, andando ad inserire un pannello per ciascuna delle operazioni descritte. In ogni pannello sono previsti un titolo, a scopo informativo, ed un pulsante per il caricamento dei dati. L'area di visualizzazione inizialmente contiene un testo che invita l'utente ad interagire con i pannelli di controllo. In seguito al caricamento dei dati, questa sezione viene popolata con le informazioni richieste e per ciascuna è presente un pulsante per la

ripresa del lavoro. La struttura risulta quindi molto semplice e di immediata comprensione per l'utente.

In seguito alla definizione del frontend, si è passati alla progettazione del **backend** per rendere operativa la pagina. Quindi, ad ogni pulsante è stata associata una funzione Python per il recupero delle informazioni richieste. Sulla base del modello di sviluppo imposto da Django e adottato dagli sviluppatori, queste funzioni devono essere poste in una o più view all'interno di un modulo. Inoltre, gli sviluppatori hanno creato ogni modulo come un'applicazione Django a sé stante. Analizzando la piattaforma, si è riscontrato come il modulo di autenticazione sia stato progettato per controllare gli accessi degli utenti ad ogni altro modulo e di come sia stato posto trasversalmente rispetto a tutte le altre componenti. Date le necessità del progetto di tesi, il modulo di sintesi deve essere collocato in parallelo rispetto alla componente dedicata all'autenticazione. L'analisi di questa soluzione ha evidenziato una sostanziale duplicazione del codice necessario per interfacciare il modulo di sintesi con le altre componenti rispetto al codice presente nel modulo di autenticazione. Si è quindi deciso di adottare una soluzione intermedia con l'obiettivo di ridurre al minimo la duplicazione di codice, ovvero sviluppare la dashboard come applicazione interna al modulo di autenticazione. Questo permette di continuare ad avere un solo modulo che si interfaccia con il resto della piattaforma al cui interno è presente l'applicazione dedicata al riepilogo delle attività. Pertanto, la dashboard è stata sviluppata in qualità di nuovo progetto Django all'interno del modulo di autenticazione.

Dal punto di vista tecnologico, la scelta del linguaggio impiegato nel backend è stata guidata dalla piattaforma stessa. È possibile sviluppare adottando linguaggi e paradigmi differenti ma si verrebbe a creare un ecosistema vario, il quale potrebbe portare ad un'instabilità del sistema e non generare benefici concreti. Inoltre, essendo la dashboard un'estensione del progetto originale, si è ritenuto opportuno utilizzare lo stesso linguaggio impiegato dal resto della piattaforma per dare continuità al software. Per quanto riguarda la parte del frontend si presentano due principali scelte progettuali, la prima consiste nel decidere se impiegare codice HTML puro oppure utilizzare un framework. La seconda scelta riguarda l'impiego di un template grafico o la creazione della struttura della pagina partendo da zero. La valutazione delle opzioni per la prima scelta ha portato alla decisione di impiegare codice HTML puro, come viene utilizzato nella versione della piattaforma oggetto di studio. Questo perché l'utilizzo di un framework comporta la necessità modificare i file di configurazione affinché venga compilata anche la parte dedicata alla dashboard. Per la compilazione del codice del framework è altresì necessario importare tutte le librerie legate al framework stesso. Oltre a ciò, è fondamentale aggiornare la

configurazione di Django per far sì che vengano utilizzati i file prodotti dalla compilazione per l'esecuzione della nuova applicazione.

La seconda scelta implementativa è ricaduta sulla creazione di una pagina senza l'impiego di un template grafico. L'utilizzo di un template comporta la necessità di adattare funzioni già esistenti per gli scopi prefissati nella propria applicazione. La configurazione è generalmente semplice ma le modifiche da apportare affinché la pagina sia quanto più simile al progetto originale comportano un dispendio di risorse notevole. Inoltre, la creazione di una pagina da zero offre una maggiore libertà di sviluppo e la riduzione del codice. I template sono sviluppati per essere il più generici possibile e cercare di soddisfare necessità diverse, pertanto l'adozione di un template comporta spesso l'utilizzo di una piccola parte delle pagine offerte, rendendo le restanti codice superfluo. Non è stato necessario intervenire sulle basi di dati in quanto il progetto di tesi è volto a mostrare informazioni opportunamente elaborate, pertanto vengono eseguite soltanto query in lettura sui dati.

5.2 Realizzazione

La dashboard si basa su un'applicazione Django creata con il comando *startapp* messo a disposizione dal framework. Questa funzione crea la struttura base di un'applicazione Django che verrà inclusa all'interno di un progetto, in questo caso il progetto è il modulo di autenticazione. Il comando viene lanciato sul terminale e va a creare, per conto del programmatore, i file base di cui un'applicazione è composta, ovvero il model e la view più alcuni file di supporto. I file relativi al template non vengono creati per dare maggior libertà allo sviluppatore di scegliere come implementare la parte client. In questo caso, si è proceduto alla creazione di una cartella apposita contenente i template impiegati dall'applicazione. La dashboard è stata sviluppata seguendo il paradigma Model View Template (MVT) utilizzato in tutti i moduli del LAS e reso disponibile dal framework Django. L'applicazione non introduce operazioni di scrittura sui database, pertanto non necessita della definizione di nuove classi all'interno del Model. Viene fatto uso esclusivamente delle classi precedentemente create nei Model del sistema e disponibili nei vari moduli interpellati.

Il nucleo elaborativo, ovvero la parte di backend, è composto da tutte le View, realizzate in Python, presenti nell'applicazione. Per renderle raggiungibili dal client, le funzioni sono mappate con determinati indirizzi URL in un file creato ad hoc. Nella View sono definite tutte le operazioni necessarie alla raccolta dei dati che verranno mostrati nella dashboard. Le funzioni ivi impiegate sono realizzate nella forma di microservizi che effettuano query di sola lettura sul database andando ad elaborare opportunamente i dati restituiti. Il frontend, effettuando richieste HTTP agli URL esposti, andrà a popolare la pagina con le informazioni utili all'utente. Per il reperimento dei dati si è proceduto alla creazione di apposite API all'interno dei moduli della Biobanca e degli Xenopazienti. Queste funzioni vanno ad utilizzare i Model già presenti nei rispettivi moduli per effettuare interrogazioni mirate ed ottenere i dati. Ciascuna funzione presente nella View richiama opportunamente le API descritte in precedenza, i risultati ricevuti vengono elaborati e poi inviati come risposta HTTP al client. Questi ultimi sono formattati in oggetti JSON per semplificare la lettura dei dati a frontend da parte del JavaScript. Il JSON (JavaScript Object Notation) è un formato testuale pensato per lo scambio di dati fra macchine. Utilizza una sintassi ridotta che lo rende un formato leggero e di facile comprensione per l'uomo, inoltre è facilmente comprensibile soprattutto per i calcolatori.

La funzione *index*, presente nella View, viene impiegata per la creazione della struttura della pagina, è la prima ad essere chiamata quando si viene reindirizzati alla dashboard. Il file è composto da diverse altre funzioni, ciascuna delle quali è associata ad un pulsante presente nel template. Ogni pulsante nel frontend è dedicato ad un'operazione specifica, pertanto ciascuna funzione è indipendente dalle altre e si serve di API differenti. Il modulo della Biobanca e degli xenopazienti predispongono di una cartella apposita destinata alle API, questa directory ospita diversi file impiegati dai rispettivi moduli per interfacciarsi ed esporre determinati servizi con gli altri moduli. Per assolvere alle necessità dell'applicazione, sono stati creati due nuovi file, uno per ciascuna cartella delle API all'interno dei moduli, ospitanti le funzioni necessarie al reperimento dei dati. L'obiettivo di queste funzioni è l'interrogazione del database attraverso le classi predefinite nei model e l'ottenimento dei dati. Anche in questo caso, ciascuna funzione si occupa esclusivamente delle informazioni relative ad una determinata procedura. Le nuove API vanno ad affiancare quelle esistenti e vengono esposte attraverso URL i cui percorsi contengono una parte in comune che ne identifica la destinazione d'impiego.

La dashboard, così come il modulo di autenticazione di cui fa parte, è quasi totalmente indipendente dal resto del sistema ad eccezione della struttura dati su cui si basa il database. Qualora vengano effettuate modifiche alla base dati e di conseguenza ai Model, sarà necessario allineare coerentemente le query eseguite dalle API riservate

all'applicazione. La View utilizza inoltre le API messe a disposizione dal MDDM per mostrare statistiche relative alle collezioni di campioni presenti nel sistema. Il MDDM è il modulo ottimizzato per l'estrazione di grandi quantità di dati presenti nell'intero database relazionale. Le interfacce esposte dal MDDM consentono di richiamare uno o più template salvati nel database dedicato al modulo e di utilizzarli per il reperimento dei dati. Il sistema dispone di una serie di template preconfigurati per le operazioni e le entità comunemente utilizzate dagli utenti. L'estrazione delle collezioni fa parte delle query presenti nel sistema, di base è configurata per estrarre i dati di tutte le raccolte di campioni presenti nel database.

È possibile eseguire queste query sia attraverso l'interfaccia grafica sia richiamando opportunamente le API esposte dal modulo MDDM. In entrambi i casi è possibile modificare la configurazione di base della query aggiungendo il vincolo che i dati siano appartenenti ad un determinato gruppo o ad un utente specifico. Si è deciso di creare una funzione nella View della dashboard che, attraverso queste API, ottenga tutte le informazioni restituite dalla query senza imporre vincoli. Questi dati vengono successivamente elaborati per mostrare all'utente le informazioni previste in fase di progettazione della pagina. Le query preconfigurate vengono memorizzate nel database del MDDM durante l'installazione del software. In Figura 3 è mostrato un esempio di come vengono rappresentate visivamente le interrogazioni al database. Per la memorizzazione di tale struttura vengono impiegate opportune stringhe JSON contenenti, fra l'altro, gli identificativi dei blocchi e l'ordine in cui devono essere collegati. La funzione che si occupa di interpretare le stringhe JSON ed eseguire le query in esse contenute, può accettare parametri in input quali il gruppo o l'utente e andare quindi a selezionare un sottoinsieme dei dati che verranno visualizzati. L'esecuzione di questa funzione restituisce a sua volta un oggetto JSON, opportunamente formattato, contenente il risultato richiesto.

La parte grafica relativa al frontend è costituita dai Template utilizzati per organizzare e mostrare il contenuto della dashboard all'utente nonché dai file JavaScript e CSS utilizzati rispettivamente per rendere dinamica la pagina e per modificare lo stile grafico della stessa. Seguendo il concetto di Template generico si è proceduto alla creazione di un Template base che definisce l'ossatura della pagina web e di un secondo Template per il contenuto vero e proprio dell'applicazione. Il corpo della pagina è strutturato in due sezioni, una di controllo ed una di visualizzazione. In seguito alla realizzazione della pagina, si è notato come l'area di controllo necessitasse di maggior spazio per ospitare i pulsanti. Inoltre, la parte di visualizzazione presentava una notevole quantità di spazio

non utilizzato. Si è pertanto deciso di modificare le proporzioni delle due parti passando da 50:50 a circa 70:30.

Come descritto in precedenza, nella prima sezione sono presenti i pulsanti con cui l'utente può richiedere le informazioni al server. Si è deciso di mostrare all'interno di questi ultimi il numero totale di attività in sospeso relativo all'entità di riferimento. Questo permette di avere immediatamente un feedback riguardo ad eventuali attività in sospeso relative all'operazione di interesse. Effettuando un click sui pulsanti la pagina interroga il server per ottenere i dati da mostrare. La sezione di visualizzazione viene impiegata per mostrare i risultati della richiesta inviata dall'utente. Nella parte dedicata alla visualizzazione, ove necessario, vengono mostrati appositi pulsanti per riprendere le attività elencate. Il funzionamento di questi pulsanti consiste nell'effettuare il reindirizzamento dell'utente alla pagina dedicata allo svolgimento della procedura selezionata.

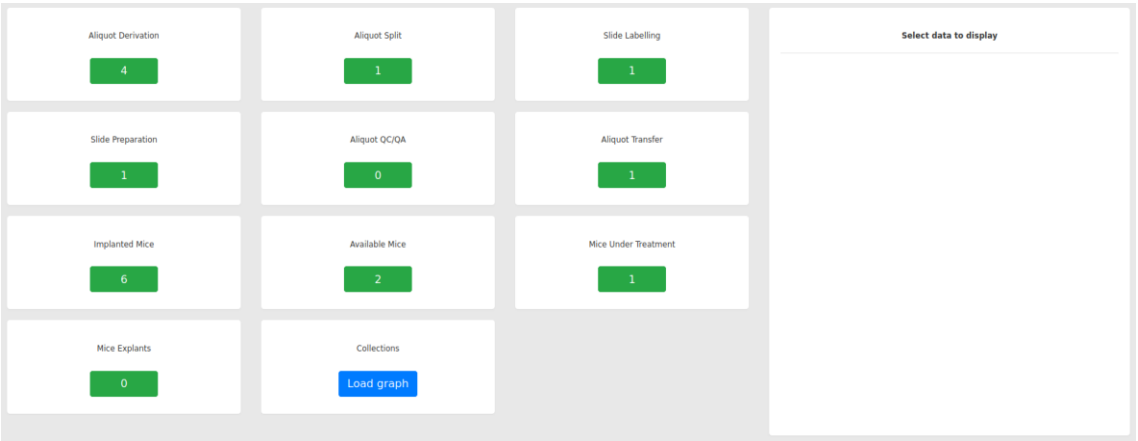

Figura 12: Corpo della pagina di riepilogo

L'analisi della versione oggetto di studio della piattaforma ha evidenziato la quasi totale mancanza di responsività nella parte grafica del software. La responsività consiste nel rendere una pagina web sempre visibile correttamente su schermi dalle diverse dimensioni. Ovvero, la pagina si adatta allo schermo su cui viene mostrata andando a modificare automaticamente la posizione degli elementi presenti. Si è deciso di sviluppare la dashboard nativamente in modalità responsive per favorirne l'utilizzo su qualunque dispositivo. Pertanto, per rendere l'applicazione compatibile dal punto di vista grafico con i numerosi dispositivi in commercio si è optato per l'utilizzo di Bootstrap [25]. Questa libreria è fra le più utilizzate al mondo per lo sviluppo di applicazioni responsive. È stata

introdotta da Twitter per diventare successivamente un progetto a sé stante con un team di sviluppo dedicato. Costantemente aggiornata, dispone di una grande quantità di temi nonché template, sia open source che proprietari, pronti all'uso. L'installazione è estremamente semplice, è possibile includere i file della libreria nel proprio progetto o più semplicemente importarli direttamente nel codice HTML attraverso i tag link e script impiegati nell'header della pagina web. Dispone delle proprie icone e ne ha recentemente introdotto una versione in formato SVG (Scalable Vector Graphics). Questa tipologia di icone è creata mediante immagini vettoriali, ovvero figure costituite da punti, linee o forme. Ciò significa una risoluzione potenzialmente infinita in quanto è possibile scalare le immagini che andranno ad adattarsi a qualunque risoluzione. Questo formato è ideale per la creazione di icone stilizzate, non è adatto ad immagini più complesse in quanto il formato vettoriale impone alcune limitazioni sul livello di dettaglio che è possibile rappresentare.

Bootstrap è quindi una libreria di strumenti open source e consente lo sviluppo di applicazioni web con grafica responsive, ovvero la pagina web si adatta alle dimensioni dello schermo su cui viene visualizzata. Bootstrap mette a disposizione dello sviluppatore numerosi identificativi, la maggior parte dei quali sono classi, contenenti il codice CSS che verrà applicato all'elemento HTML. Il framework è basato su un sistema che suddivide la pagina in una griglia composta da dodici colonne. Questo semplifica notevolmente lo sviluppo permettendo di scrivere codice che si adatta sia su dispositivi mobile che desktop. Le colonne, nel momento in cui lo schermo sia di dimensioni ridotte, si distribuiscono automaticamente su una o più righe sottostanti andando ad occupare lo spazio verticalmente.

Inoltre, gli sviluppatori della libreria hanno individuato cinque possibili suddivisioni per i dispositivi ad oggi in commercio basandosi sulla grandezza dello schermo di cui questi ultimi sono dotati. Per ciascuna proprietà grafica trattata nella libreria sono quindi presenti cinque identificativi, uno per ogni suddivisione individuata. Il browser, interpretando il codice JavaScript compreso nella libreria, dispone di conseguenza gli elementi nella pagina web. Qualora lo sviluppatore abbia inserito un identificativo per ogni suddivisione, il JavaScript applica il codice CSS opportunamente effettuando un adattamento in tempo reale. Questo tipo di adattamento viene adottato altresì nel caso in cui si utilizzi un dispositivo mobile ed è maggiormente visibile qualora si cambi l'orientamento del sistema in uso. La progressiva sostituzione dei terminali fissi con computer portatili o tablet dallo schermo ridotto rendono questi strumenti obbligatori per offrire la maggior compatibilità possibile con l'hardware in commercio. La pagina web risulta quindi fruibile a prescindere dal dispositivo su cui viene visualizzata. L'utilizzo di

questa libreria o di prodotti simili è divenuto un requisito necessario per qualsiasi piattaforma online.

Il nucleo operativo del frontend fa uso del JavaScript, nello specifico della libreria **jQuery**, per attivare i pulsanti, inviare le richieste al server ed elaborare le risposte ricevute. Le chiamate al server vengono effettuate in modo asincrono col costrutto AJAX (Asynchronous JavaScript And XML) rendendo l'aggiornamento della pagina dinamico senza necessità di ricaricamenti. Ciascun pulsante presente nell'area di controllo dispone della propria funzione JavaScript che viene attivata al click dell'utente. Ognuna di queste funzioni si collega al server mediante chiamata AJAX per reperire i dati e popolare la parte relativa alla visualizzazione con le informazioni ricevute. Il costrutto AJAX è stato implementato inserendo due delle possibili callback messe a disposizione da jQuery, ovvero le funzioni success ed error. La callback riservata al success viene chiamata in caso di risposta positiva da parte del server e si occupa di validare i dati e costruire il codice HTML che verrà mostrato nella sezione dedicata alla visualizzazione.

La validazione dei dati consiste nel verificare un parametro inserito appositamente lato server per segnalare eventuali eccezioni scatenate dal codice eseguito nel backend. Questo controllo ha uno scopo orientato prevalentemente alla fase di sviluppo e test delle funzioni. In caso di eccezione il parametro viene valorizzato con il messaggio restituito dall'errore ed è visibile nella console per gli sviluppatori. La callback error, dedicata ad eventuali errori, viene quindi attivata esclusivamente in caso di errori più complessi accorsi nel server o nella comunicazione tra frontend e backend. Nella funzione riservata al success viene composto programmaticamente il codice HTML in base ai dati ricevuti dal server. La stringa HTML ottenuta va a sostituire il contenuto del corpo dell'area di visualizzazione.

Per ogni chiamata eseguita verso il server, durante l'attesa della risposta è stato inserito uno spinner di caricamento per informare l'utente che la pagina sta effettuando delle elaborazioni. Le statistiche relative alle collezioni presenti nella piattaforma vengono mostrate mediante un grafico a torta. Per la creazione del grafico si è deciso di utilizzare la libreria JavaScript **chart.js** [26] in quanto open source, responsive e compatibile con la maggior parte dei browser in commercio. La libreria necessita del tag *canvas*, introdotto con la versione 5 di HTML come contenitore generico. All'interno di questo tag è possibile disegnare grafici o oggetti più complessi mediante appositi script. Chart.js mette a disposizione un oggetto JavaScript creato ad hoc mediante il quale è possibile definire i parametri e i dati che il grafico andrà a mostrare. Ogni oggetto JavaScript deve corrispondere ad uno specifico grafico, nel caso della dashboard è presente un solo oggetto correlato al diagramma. L'inizializzazione dell'oggetto permette alla libreria di

identificare il diagramma e predisporre opportunamente lo spazio all'interno del canvas. Fra i parametri è possibile selezionare la tipologia di grafico che verrà visualizzato, personalizzare la legenda così come le didascalie sugli assi dei diagrammi di tipo cartesiano qualora vengano mostrati.

La tipologia di diagrammi a torta impiegata permette di assegnare un colore diverso ad ogni spicchio del diagramma. Per l'assegnazione dei colori si è utilizzata una funzione che genera codici numerici casuali i quali verranno convertiti in codici rgb per ottenere i colori. Pertanto, nei grafici ove previsto, è disponibile il parametro di configurazione relativo ai colori che accetta un vettore di codici colore in formato rgb. Il pulsante per la visualizzazione del diagramma richiama la funzione dedicata al recupero dei dati. Questa funzione interroga il backend e aggiorna l'oggetto contenente il grafico inserendo le informazioni ricevute e generando i codici destinati ai colori. Successivamente viene invocato l'apposito metodo update che andrà ad aggiornare i dati del diagramma visualizzato.

La libreria necessita di un tag canvas all'interno della pagina web che sia presente fin dalla fase di caricamento della pagina stessa. Questo per poter associare l'oggetto JavaScript al tag ed andare a configurare opportunamente il diagramma. Si è deciso quindi di inserire una sezione nell'area di visualizzazione dedicata esclusivamente al diagramma. Questa parte viene mostrata nel momento in cui l'utente richiama la funzione di visualizzazione del grafico, in tutti gli altri casi risulta non visibile. Per l'impiego della libreria si è utilizzato la modalità CDN (Content Delivery Network), ovvero viene effettuato il download della libreria durante il caricamento iniziale della pagina per poi essere disponibile per tutta la sessione di lavoro. Questa modalità di impiego permette di aver accesso a runtime alla versione online della libreria. Pertanto, non è necessario integrare i vari file di cui è composta nel progetto dell'applicazione andando così ad alleggerire il software.

Gli stili grafici così come le animazioni presenti nella dashboard sono definiti nel file CSS dedicato all'applicazione. Per apportare le modifiche CSS è stato privilegiato l'uso delle classi, rispetto agli id. Pertanto, vengono impiegate le classi come identificatori per l'applicazione degli stili in quanto più generiche e facilmente riutilizzabili all'interno del progetto. Le animazioni e la responsività del diagramma derivano dal CSS integrato all'interno della libreria chart.js. L'intera struttura della dashboard è realizzata applicando classi fornite da Bootstrap. La sezione di controllo è strutturata con una griglia avente tre colonne e tre righe e contenente un pulsante per ogni casella. L'area di visualizzazione fa uso di una sola colonna e di un numero variabile di righe determinato da tutti i possibili step in cui l'operazione selezionata è composta. Su schermi ridotti si è deciso di

visualizzare l'area di controllo nella parte superiore dello schermo mentre la sezione di visualizzazione viene posta nella parte inferiore.

5.2.1 Miglioramenti apportati

Nel corso dello sviluppo della dashboard sono stati effettuati alcuni cambiamenti al progetto iniziale per migliorare le prestazioni ed aggiungere funzionalità all'applicativo. Una prima versione della dashboard prevedeva il caricamento delle informazioni da parte del server prima della visualizzazione della pagina. Al termine del caricamento la pagina veniva restituita al client e risultava quindi popolata fin da subito con i dati. Questo richiedeva un tempo alto di caricamento iniziale dovuto all'esecuzione di tutte le query necessarie per avere a disposizione la pagina e poter operare. Inoltre, il tempo impiegato dalle query tende a crescere con l'aumentare dei dati trattati, pertanto la soluzione inizialmente adottata non era soddisfacente dal punto di vista delle prestazioni. La fruibilità dell'applicativo risultava bloccata in attesa di una risposta da parte del backend, per un tempo pari alla somma dei tempi delle operazioni eseguite.

Come soluzione a questo problema si è optato per ridurre al minimo i caricamenti iniziali effettuati dal server e restituire immediatamente al client una prima versione della pagina popolata con informazioni temporanee. La compilazione dei dati nella schermata viene eseguita lato client attraverso alcune chiamate asincrone effettuate in JavaScript. Si è quindi deciso di effettuare lo scaricamento dei dati in tempo reale, non appena viene visualizzata la pagina, andando ad applicare stili diversi ai controlli per informare l'utente che l'applicazione sta effettuando delle operazioni. Al termine dello scaricamento dei dati, le informazioni temporanee vengono progressivamente rimpiazzate dai dati veri e propri. Per far ciò, sono state inserite funzioni JavaScript ad hoc che si occupano del primo caricamento dei dati. Queste operazioni vanno a richiamare apposite procedure nel backend che restituiscono esclusivamente il valore che verrà inserito come testo nel pulsante. Le operazioni eseguite nel backend sono le medesime che vengono lanciate effettuando click su ciascuno dei pulsanti, pertanto vengono impiegate le stesse API. La differenza sostanziale risiede nei valori restituiti al client, le nuove funzioni vanno a sommare tutti gli indici ottenuti dalle API per mostrare il testo con il valore corretto all'interno del pulsante.

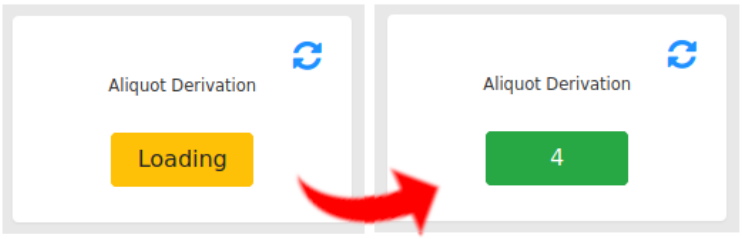

Figura 13: Esempio di primo caricamento

La Figura 13 mostra come l'utente venga informato sulle operazioni eseguite dal sistema mediante feedback visivo. Inoltre, al termine del caricamento può dedurre immediatamente quante operazioni sono rimaste incompiute. In questo caso non viene visualizzato lo spinner di caricamento dato che alcune operazioni potrebbero terminare prima di altre e quindi permettere all'utente di interagire immediatamente con i pulsanti attivi. In una prima fase di creazione della pagina, è quindi possibile visualizzare alcuni pulsanti con testo "Loading" e altri col valore correttamente compilato. Questo è dovuto principalmente alla natura delle chiamate asincrone, le quali hanno tempi diversi di esecuzione con conseguente restituzione dei risultati a distanza di qualche istante una dall'altra. Queste modifiche aggiungono dinamicità alla pagina oltre a rendere l'utente conscio di ciò che sta accadendo.

Effettuando alcuni test di utilizzo della dashboard si è notato che, nel momento in cui l'utente intende terminare un'operazione lasciata in sospeso, si ha un reindirizzamento alla pagina di esecuzione corretta. Al termine dell'operazione, qualora l'utente intenda terminarne un'altra, è necessario tornare al menu principale per poter accedere nuovamente alla dashboard. Questo perché il reindirizzamento avveniva inizialmente nella stessa scheda del browser. Pertanto, per offrire un'esperienza d'uso migliore e incrementare la produttività degli utenti, si è deciso di effettuare il reindirizzamento in una nuova scheda. Questo consente di portare a termine l'operazione e di ritornare immediatamente alla dashboard per riprendere altre procedure. Con questa modalità operativa la dashboard diventa una sorta di punto di partenza per l'esecuzione di queste e altre operazioni, mantenendo la scheda aperta e sfruttando i link per accedere direttamente alla procedura nel modulo di interesse.

Questa nuova tipologia di utilizzo ha introdotto un'altra necessità, derivante dalle modifiche effettuate ai dati. Dal momento in cui la pagina di riepilogo viene caricata, non c'è modo di effettuare l'aggiornamento dei dati in essa mostrati se non ricaricando

completamente la pagina web. Si ha quindi la visualizzazione di dati non più allineati con quanto presente nei database a seguito delle modifiche. Per ovviare a questa necessità sono state individuate alcune possibilità che consistono nell'inserire un task di aggiornamento in background, un pulsante di caricamento dell'intera pagina o un pulsante per ogni singolo pannello dell'area di controllo. La prima soluzione risulta troppo onerosa in termini di prestazioni, richiede di inserire un tempo di timeout né troppo lungo, né troppo breve. Inoltre, è possibile che questa pagina venga lasciata aperta su uno o più terminali e che questo generi traffico eccessivo per il server.

Le altre soluzioni proposte sono entrambe valide e sono dipendenti l'una dall'altra. Pertanto, al fine di aggiornare le informazioni presenti nella videata sono state implementate queste due modalità, la prima prevede uno scaricamento completo dei dati presenti mentre la seconda consente nell'aggiornamento di una singola sezione alla volta. L'unione di queste soluzioni garantisce all'utente la massima libertà decisionale oltre a non comportare codice aggiuntivo. Quindi, è stato implementato un pulsante apposito per effettuare nuovamente lo scaricamento di tutti i dati senza dover ricaricare l'intera pagina. Mentre la seconda opzione prevede un'icona per ogni pulsante nella sezione di controllo che consente di effettuare un aggiornamento mirato dei dati. In entrambe le modalità, il sistema reimposta i dati temporanei andando a modificare lo stile della pagina per evidenziare l'aggiornamento in corso. Il pulsante per l'aggiornamento di tutti i dati richiama la stessa funzione impiegata per il primo accesso alla pagina. Questa funzione è composta dalle chiamate alle procedure di caricamento dei dati di ogni singola sezione. Al contrario, l'icona per lo scaricamento mirato dei dati richiama esclusivamente la funzione dedicata alla sezione che si vuole aggiornare.

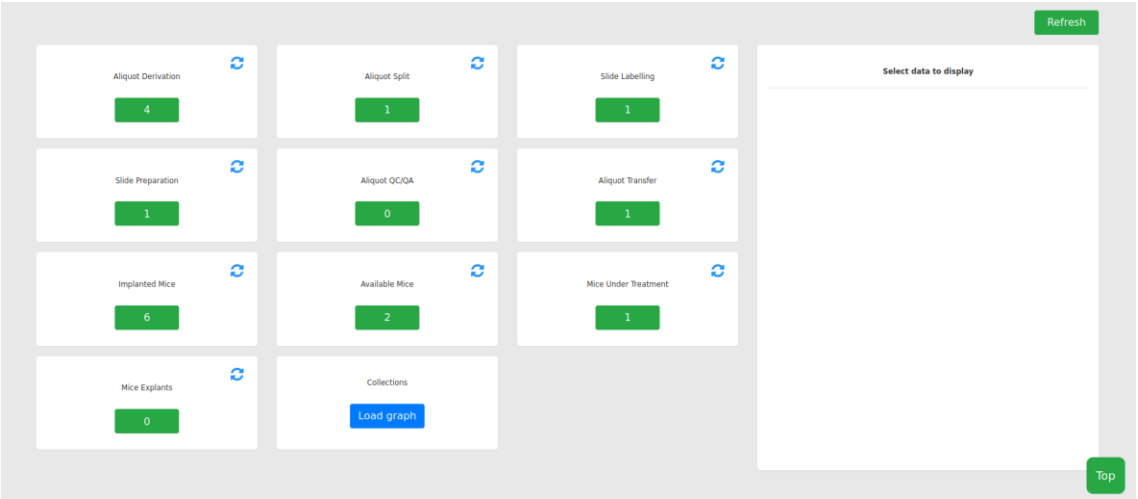

Figura 14: Grafica dashboard a seguito degli aggiornamenti

In Figura 14 viene mostrato un ulteriore miglioramento apportato alla pagina. Effettuando test con uno schermo di dimensioni ridotte, si è notato come l'utente debba continuamente effettuare uno scorrimento verso il basso per visualizzare i dati caricati a seguito del click su uno dei pulsanti di controllo. Si è deciso di inserire uno scorrimento automatico della pagina verso il basso, qualora venga rilevato l'utilizzo di uno schermo di dimensioni ridotte. Inoltre, per riportare velocemente l'utente ad inizio pagina, si è proceduto all'inserimento di un pulsante a comparsa che rimanda nuovamente in testa alla videata. Dal punto di vista del backend si è riscontrata la presenza di una serie di funzioni il cui scopo è interrogare le API e restituire i dati ricevuti senza apportare alcuna modifica significativa. Si è pertanto deciso di far puntare le chiamate del frontend direttamente alle rispettive funzioni di origine delle API, andando a rimuovere le procedure all'interno della View ed evitando un passaggio inutile.

6 Casi d'uso

I casi d'uso possono essere definiti come lo studio degli scenari di operatività di un sistema e degli utilizzatori in essi coinvolti. Rappresentano quindi le funzionalità messe a disposizione dal sistema per i suoi utilizzatori. Questi modi in cui un sistema può essere utilizzato vengono descritti mediante scenari con interazioni fra gli utilizzatori ed il sistema oggetto di studio. Non vengono specificate le attività interne svolte dal sistema, il caso d'uso è incentrato sull'interazione fra l'utente ed il sistema, quindi può essere considerato come una scatola nera. Inoltre, deve essere il più elementare possibile, ovvero non essere scomponibile in altri casi d'uso più semplici. I casi d'uso vengono generalmente impiegati per la definizione dei requisiti funzionali di un sistema e per la successiva validazione e verifica dello stesso. Vengono descritti mediante una sequenza di azioni con lo scopo di produrre un risultato osservabile da uno o più attori coinvolti. Questa sequenza di azioni delinea l'interazione fra entità esterne al sistema, ovvero gli attori, ed il sistema stesso o una parte di esso. Fanno parte degli attori tutte le persone fisiche o altri sistemi esterni all'apparato oggetto di studio che interagiscono col sistema. Gli attori eseguono i casi d'uso e ne ottengono una qualche forma di valore, non fanno quindi parte del sistema ma scambiano informazioni con esso. Sono connessi ai casi d'uso mediante associazioni o relazioni di comunicazione, pertanto sia l'attore che il sistema possono inviare o ricevere messaggi reciprocamente fra di loro.

Il principale attore che interagisce con la dashboard è l'utente, sotto forma di ricercatore, che impiega la pagina di riepilogo per visualizzare lo stato delle lavorazioni e riprendere le attività. La piattaforma dispone di altre tipologie d'utenza, il Principal Investigator (P.I.) e gli amministratori. Questi ultimi, come visto in precedenza, hanno compiti di controllo e non hanno accesso alla biobanca pertanto non possono accedere alla dashboard. Gli utenti di tipo P.I. non dispongono di privilegi particolari per quanto riguarda la pagina di riepilogo, quindi vengono considerati utenti normali e si applicano gli stessi casi d'uso validi per gli utenti base. Non sono presenti altre tipologie di utilizzatori legati alla pagina di riepilogo, pertanto l'unico attore individuabile è la persona fisica che può accedere alla dashboard ed operare sul sistema. Per lo sviluppo della dashboard ci si è basati sulle richieste dell'utenza nonché sui progetti di ampliamento stabiliti dagli sviluppatori, i casi d'uso seguenti hanno lo scopo di validare la struttura del nuovo componente e verificarne il corretto funzionamento. A seguito

dell'identificazione degli attori, si è proceduto con l'identificazione dell'interazione principale e di eventuali eccezioni agli scenari. L'interazione, che è comune alla maggior parte degli scenari considerati, consiste nel mostrare all'utente una serie di dati e di permettergli di navigare alla pagina relativa alla procedura scelta.

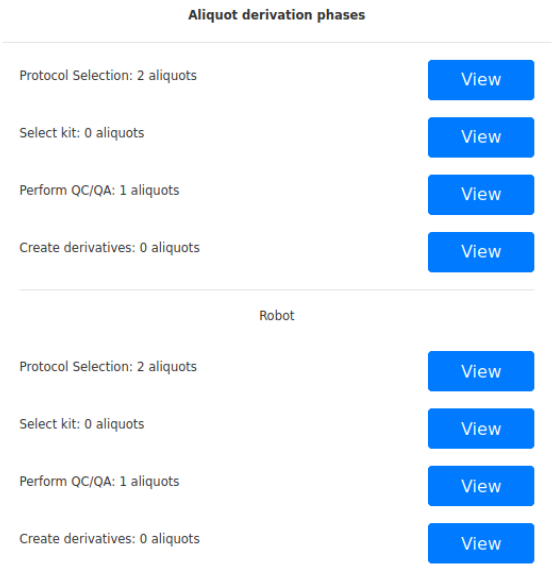

Figura 15: Esempio di step manuali e robot relativi alla derivazione

Le eccezioni si presentano per tutte quelle operazioni che possono essere eseguite sia manualmente dall'utente sia mediante l'impiego di robot. Per queste procedure il sistema deve mostrare anche il pulsante di reindirizzamento alla pagina per l'esecuzione della stessa mediante robot. Dal punto di vista tecnico è quindi possibile individuare uno scenario diverso per ciascun pulsante presente nella pagina, il quale si ramifica nel caso in cui la procedura selezionata abbia la possibilità di essere eseguita mediante robot. Inoltre, è necessario considerare le operazioni, come la derivazione mostrata in Figura 15, che vengono eseguite in step diversi e danno origine ad uno scenario diverso per ciascuna fase di cui sono composte.

Analizzando la struttura della pagina di riepilogo è possibile raggruppare i casi d'uso basandosi sul **modulo di destinazione** dell'interazione. Quindi, sono stati individuati tre differenti gruppi di casi d'uso legati all'interazione comune ed alle nuove funzionalità introdotte a seguito dei test. Pertanto, gli elementi del primo gruppo rimandano tutti alla

Biobanca, il secondo effettua un reindirizzamento al modulo degli xenopazienti, il terzo gruppo rappresenta tutti quei pulsanti che non effettuano un reindirizzamento all'esterno della dashboard. In quest'ultimo gruppo, abbiamo i casi d'uso relativi ai pulsanti di ricaricamento di ogni singolo pannello, dell'intera pagina di riepilogo ed il pulsante di visualizzazione delle statistiche.

Di seguito si procederà con l'analisi dei principali casi d'uso individuati. Verranno analizzati i casi appartenenti a ciascuno dei tre gruppi precedentemente elencati mediante l'uso di una descrizione delle azioni eseguite dall'utente. Inoltre, la relazione elencherà altresì tutte le informazioni restituite o le azioni compiute dal sistema in risposta all'input ricevuto dall'utente.

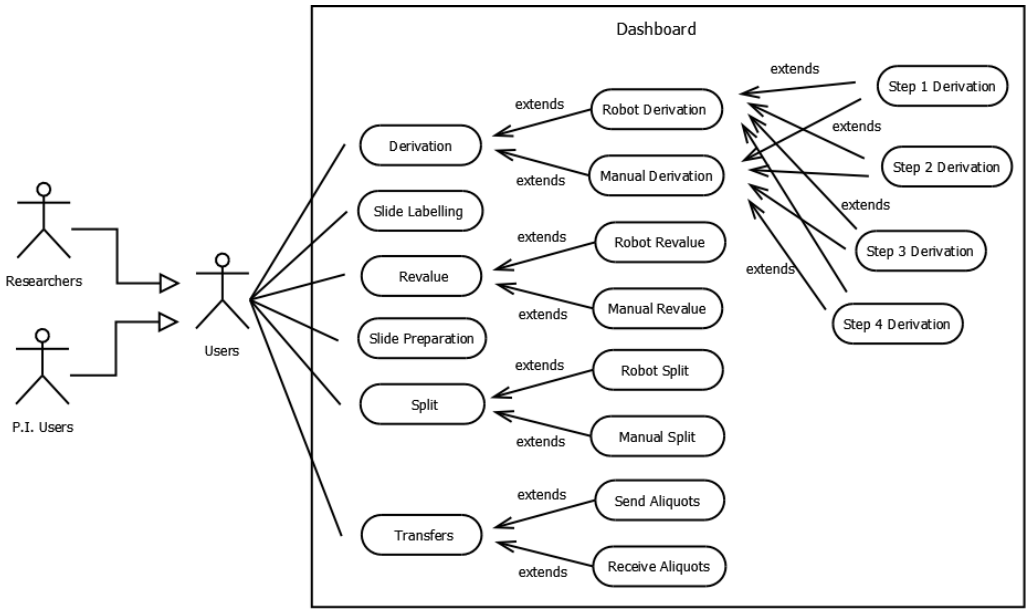

Figura 16: Diagramma UML dei casi d'uso relativi al modulo della Biobanca

Il diagramma in Figura 16 mostra l'elenco dei casi d'uso relativi al modulo della Biobanca, andando a rappresentare il primo raggruppamento delle operazioni descritto in precedenza. È possibile individuare nella prima colonna i casi d'uso principali relativi ai pulsanti presenti nei pannelli di controllo, questi estendono eventuali loro flussi alternativi derivanti dalla possibilità di effettuare operazioni in manuale o mediante robot.

6.1 Derivazione

Questo caso d'uso è relativo all'operazione di derivazione eseguibile all'interno del modulo della Biobanca. La precondizione affinché il caso d'uso sia valido è che l'utente possa avere accesso alla dashboard e che stia visualizzando la pagina di riepilogo.

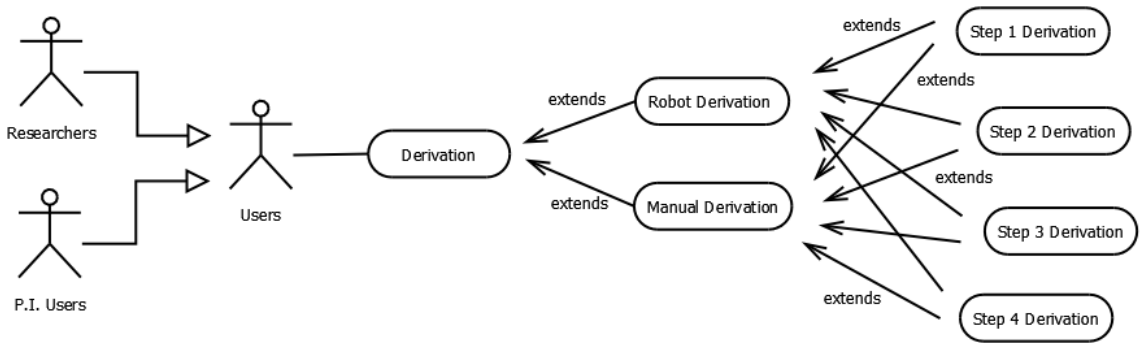

Figura 17: Diagramma UML relativo all'operazione di derivazione

A questo punto, il flusso principale degli eventi prevede che l'utente clicchi sul pulsante nel pannello relativo alla derivazione all'interno dell'area di controllo. Il pulsante contiene il numero di aliquote che necessitano dell'attenzione dell'utente. Il sistema mostra i pulsanti e le descrizioni per le fasi eseguibili manualmente e con robot all'interno dell'area di visualizzazione, come mostrato in Figura 15. Per ciascuna descrizione abbiamo il dettaglio delle aliquote coinvolte in quel determinato step. Successivamente, l'utente clicca sul pulsante di reindirizzamento relativo allo step della derivazione desiderato. Il sistema procede ad effettuare il caricamento della pagina relativa allo step della derivazione selezionato.

La post condizione, ovvero la fine naturale del flusso degli eventi, è la visualizzazione da parte dell'utente della pagina relativa allo step della derivazione selezionato. Il caso d'uso preso in considerazione, presenta due possibili flussi alternativi, i quali non portano alla conclusione naturale prevista. Il primo flusso alternativo è rappresentato dalla volontà dell'utente di abbandonare la pagina di riepilogo. Questo flusso può aver luogo in qualunque momento ed è a completa discrezione dell'utente. Il secondo flusso alternativo prende in considerazione l'eventualità di un errore del sistema. Nel caso si verifichi un errore, il sistema non potrà caricare il valore corretto all'interno del pulsante nel pannello

di controllo, pertanto mostrerà la dicitura “Loading”. Qualora il sistema non sia raggiungibile, sono previsti due possibili scenari. Nel caso in cui l’utente abbia caricato in precedenza la pagina di riepilogo, i testi dei pulsanti presenti nei pannelli di controllo conterranno la dicitura “Loading” e i pulsanti di redirect non effettueranno gli indirizzamenti. Il secondo scenario prevede che l’utente non abbia ancora effettuato l’accesso alla pagina e si conclude immediatamente in quanto quest’ultima non verrà caricata correttamente.

6.2 Slide Labelling

Il caso d’uso in questione è relativo all’operazione di etichettatura dei vetrini eseguibile all’interno del modulo della Biobanca. La precondizione è comune al caso d’uso precedentemente descritto.

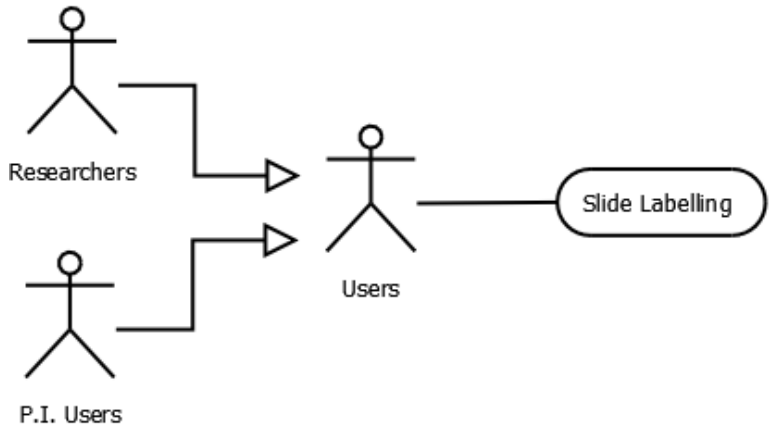

Figura 18: Rappresentazione UML specifica per Slide Labelling

Lo svolgimento di questo caso presuppone che l’utente clicchi sul pulsante nel pannello relativo allo slide labelling all’interno dell’area di controllo. Il pulsante contiene il numero di aliquote destinate all’operazione di slide labelling come sottolineato in Figura 19. Il sistema mostra il pulsante di reindirizzamento e la descrizione relativi all’operazione di

slide labelling all'interno dell'area di visualizzazione. L'utente procede cliccando sul pulsante di redirect. Il sistema effettua il caricamento della pagina di esecuzione dello slide labelling. Il caso d'uso si conclude con l'utente che visualizza la pagina relativa all'operazione di slide labelling. I possibili flussi alternativi sono i medesimi visti per l'operazione di derivazione.

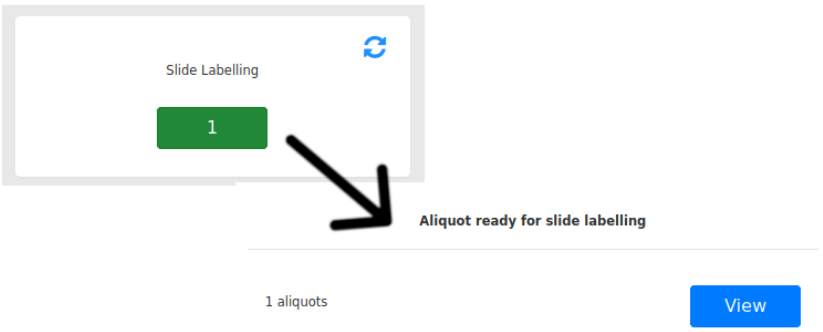

Figura 19: Dettaglio pannello e area di visualizzazione per slide labelling

6.3 Rivalutazione

Questo caso d'uso tratta la rivalutazione di un'aliquota derivata, operazione eseguibile anch'essa all'interno del modulo della Biobanca. La precondizione ricalca quanto elaborato nei precedenti casi d'uso, ovvero è necessario che l'utente possa accedere e stia attualmente visualizzando la dashboard.

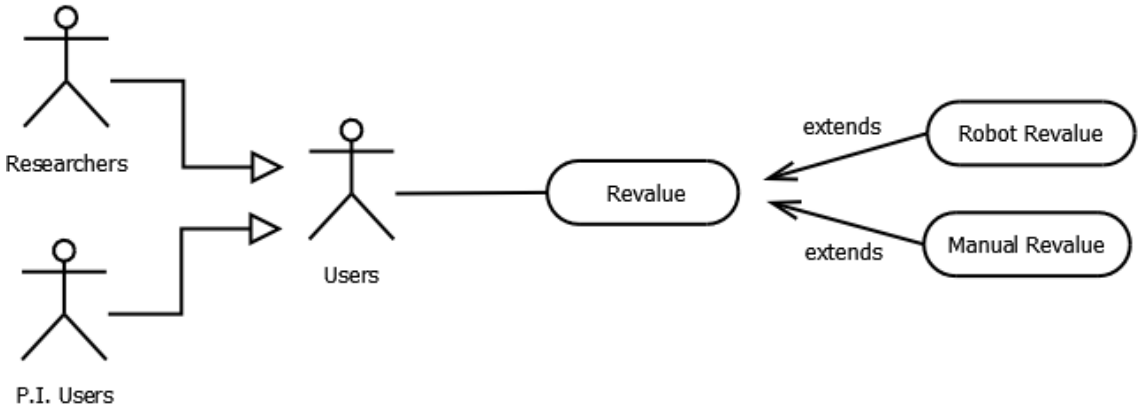

Figura 20: Dettaglio diagramma UML relativo all'operazione di rivalutazione

L'utente procederà quindi effettuando un click sul pulsante nel pannello relativo alla rivalutazione all'interno dell'area di controllo. Anche in questo caso il pulsante contiene l'indicazione di quante aliquote devono essere sottoposte a rivalutazione. Il sistema mostra, all'interno dell'area di visualizzazione, i pulsanti di reindirizzamento e le descrizioni relativi all'operazione di rivalutazione eseguibile manualmente o mediante robot. Le descrizioni conterranno lo stesso numero di aliquote in quanto durante la procedura di schedulazione dell'operazione, l'utente non può ancora selezionare la modalità di esecuzione. L'utente clicca sul pulsante di reindirizzamento relativo alla modalità di esecuzione desiderata. Il sistema effettua il caricamento della pagina relativa alla rivalutazione eseguibile secondo la modalità scelta dall'utente. Il caso d'uso termina con la visualizzazione da parte dell'utente della pagina di esecuzione dell'operazione di rivalutazione. Anche per questo caso d'uso l'utente può abbandonare la pagina in qualunque momento ed è possibile che si verifichino gli errori precedentemente descritti. Entrambi i flussi alternativi verranno gestiti al pari dei casi d'uso finora elaborati.

6.4 Slide Preparation

Il caso d'uso in questione è relativo all'operazione di preparazione dei vetrini eseguibile all'interno del modulo della Biobanca. La precondizione impone che l'utente abbia accesso e stia visualizzando la pagina della dashboard.

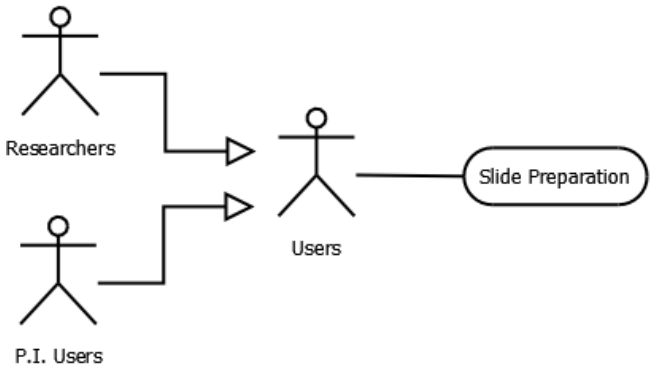

Figura 21: Rappresentazione UML specifica per Slide Preparation

L'utente inizia il caso d'uso cliccando sul pulsante nel pannello relativo all'operazione di slide preparation all'interno dell'area di controllo. Il pulsante mostra quante aliquote sono state schedulate per l'operazione. Il sistema compila la parte di visualizzazione mostrando il pulsante di reindirizzamento e la descrizione per l'esecuzione in manuale dell'operazione di slide preparation. La procedura prevede attualmente un'unica modalità di esecuzione. In seguito, l'utente clicca sul pulsante di reindirizzamento precedentemente mostrato. Il sistema procede col caricamento della pagina di esecuzione dell'operazione di slide preparation. La conclusione naturale di questo caso d'uso è rappresentata dall'utente che visualizza la pagina relativa all'operazione di slide preparation all'interno del modulo della Biobanca, come mostrato in Figura 22. I flussi alternativi seguono le interazioni viste in precedenza e vengono gestiti in egual modo.

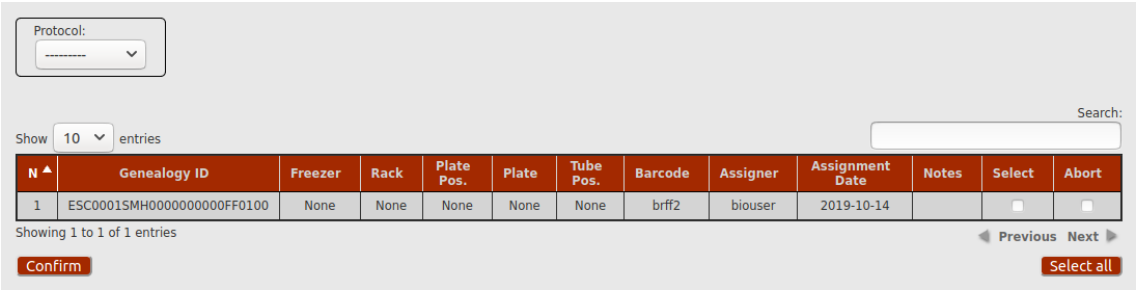

Figura 22: Post condizione per slide preparation

6.5 Split

Il caso d'uso di seguito descritto tratta l'operazione di split eseguita su aliquote di tipo derivato all'interno della Biobanca. La precondizione, anche per questo caso, è la possibilità di accedere alla pagina di riepilogo da parte dell'utente. Inoltre, l'utente deve visualizzare attualmente la dashboard.

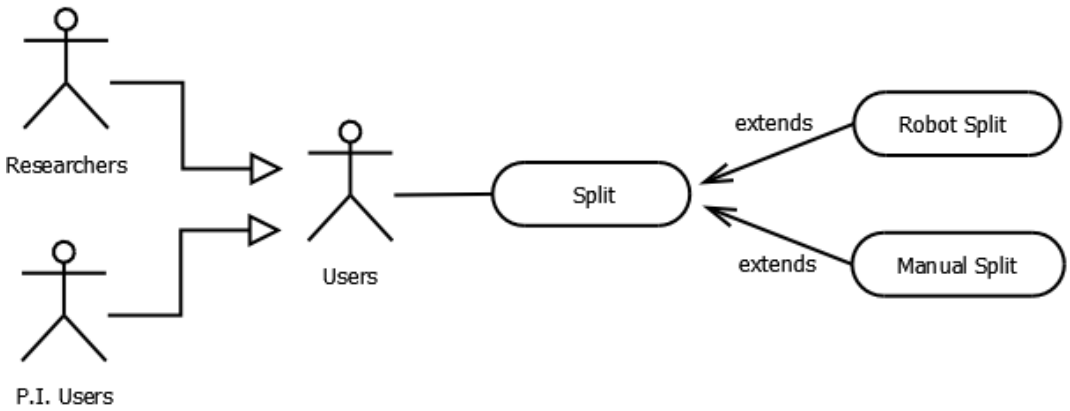

Figura 23: Diagramma UML rappresentante lo Split

A questo punto, l'utente dovrà cliccare sul pulsante nel pannello relativo all'operazione di split presente nell'area di controllo. Il sistema mostrerà, nell'area di visualizzazione, i

pulsanti e le descrizioni per l'accesso alla pagina di esecuzione dell'operazione di split in modalità manuale o mediante robot. Anche in questo caso le descrizioni mostreranno lo stesso numero di aliquote in quanto la schedulazione non prevede la scelta della modalità di esecuzione. L'utente seleziona il pulsante di reindirizzamento relativo alla modalità di esecuzione desiderata.

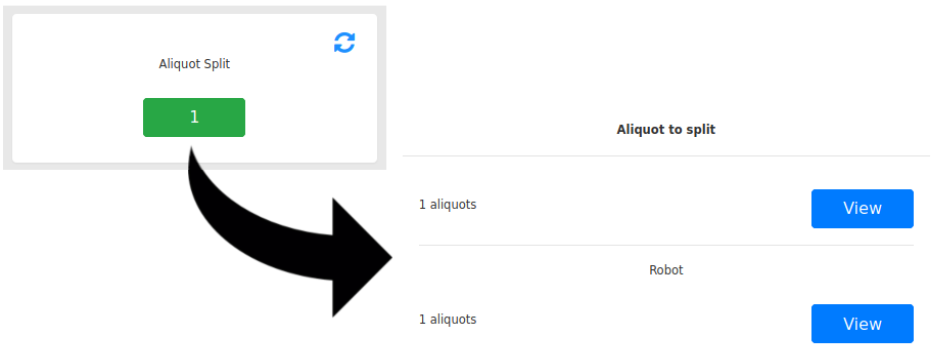

Figura 24: Caricamento dell'area di visualizzazione

Il sistema procederà quindi al caricamento della pagina di esecuzione dell'operazione di split secondo la modalità scelta dall'utente. Questa rappresenta la normale condizione di terminazione del caso d'uso. Le condizioni di terminazione anomale risultano essere le stesse dei casi d'uso visti in precedenza.

6.6 Transfers

L'ultimo caso d'uso del gruppo relativo al modulo della Biobanca prende in considerazione i trasferimenti di aliquote. La precondizione relativa a questo caso d'uso è la stessa dei casi precedenti.

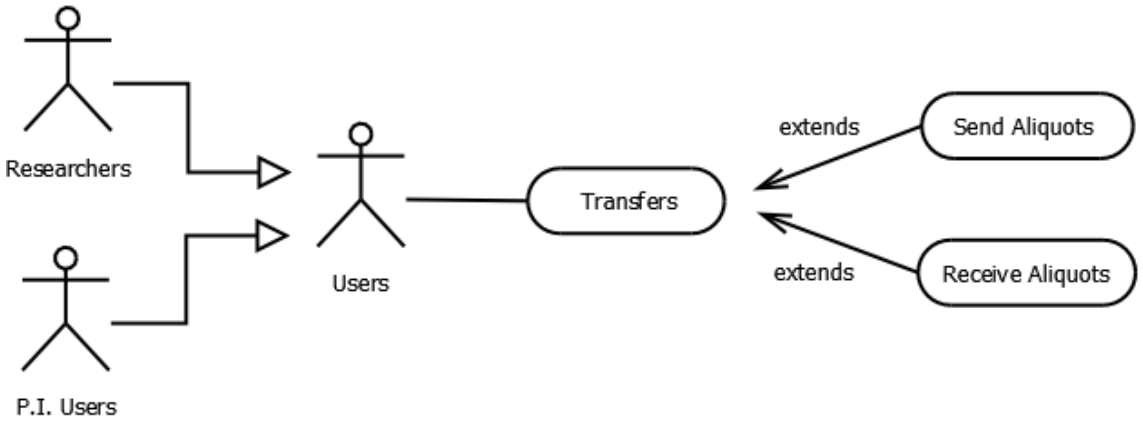

Figura 25: Dettaglio diagramma UML dei trasferimenti delle aliquote

L'utente procede selezionando il pulsante nel pannello all'interno dell'area di controllo relativo ai trasferimenti. Il pulsante mostra il totale delle aliquote che devono essere ricevute sommato al totale delle aliquote che devono essere inviate. Il sistema mostra i pulsanti di reindirizzamento e le descrizioni per le operazioni di invio e di ricezione delle aliquote all'interno dell'area di visualizzazione. In seguito, l'utente seleziona l'operazione di suo interesse ed il sistema procede col reindirizzamento alla pagina relativa all'operazione scelta. Il caso d'uso termina con l'utente che visualizza la pagina dedicata all'invio o alla ricezione delle aliquote all'interno del modulo della Biobanca. I possibili flussi alternativi legati a questo caso d'uso ricalcano quanto descritto nei precedenti casi.

Il secondo gruppo di casi d'uso è anch'esso individuabile mediante il modulo di destinazione comune, ovvero il modulo relativo agli Xenopazienti. Il diagramma in Figura 26 mostra come questa categoria di casi d'uso non abbia estensioni in quanto tutte le operazioni vengono eseguite manualmente. Inoltre, le procedure sono composte da un flusso di esecuzione unico, che non prevede possibilità di interruzioni durante lo sviluppo dello stesso.

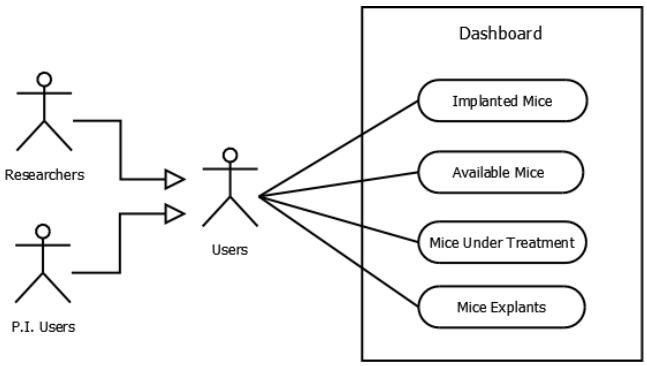

Figura 26: Diagramma UML dei casi d'uso relativi al modulo degli Xenopazienti

6.7 Implanted Mice

Il caso d'uso in considerazione permette all'utente di verificare lo stato dei topi impiantati all'interno del modulo degli Xenopazienti. La precondizione risulta essere la medesima analizzata nei casi precedenti. L'utente dovrà quindi appartenere alle tipologie abilitate all'accesso e visualizzare la pagina della dashboard.

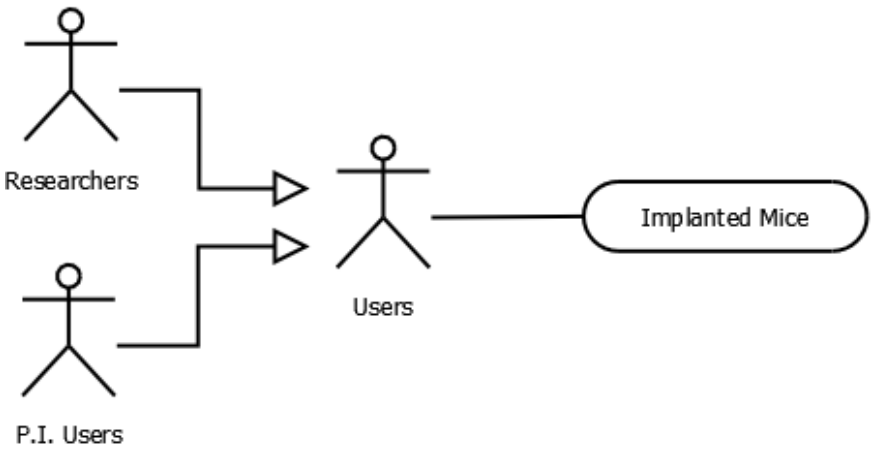

Figura 27: Rappresentazione UML relativa agli Xenopazienti impiantati

L’utente procede cliccando sul pulsante nel pannello relativo ai topi impiantati all’interno dell’area di controllo. Anche in questo caso il pulsante contiene l’indicazione di quanti animali sono in stato impiantato. Il sistema mostra la descrizione col numero di animali impiantati ed un pulsante per il reindirizzamento al modulo degli Xenopazienti. L’utente seleziona il pulsante di redirect, il sistema effettua quindi il caricamento della schermata contenente l’elenco dei gruppi a cui appartengono i topi impiantati. La post condizione è soddisfatta nel momento in cui l’utente viene correttamente indirizzato alla pagina di consultazione dei topi all’interno del modulo degli Xenopazienti. Le condizioni di terminazione non comuni rimangono invariate rispetto agli altri casi d’uso. Pertanto, l’utente può decidere di abbandonare in qualunque momento la pagina della dashboard. Inoltre, in caso di errore, il sistema risulterà parzialmente o totalmente inutilizzabile come analizzato in precedenza.

6.8 Available Mice

Questo caso d’uso tratta anch’esso l’accesso al modulo degli Xenopazienti andando a mostrare all’utente quanti topi sono disponibili per l’operazione di impianto. La precondizione risulta essere la possibilità da parte dell’utente di accedere alla pagina della dashboard e che si trovi attualmente nella suddetta pagina.

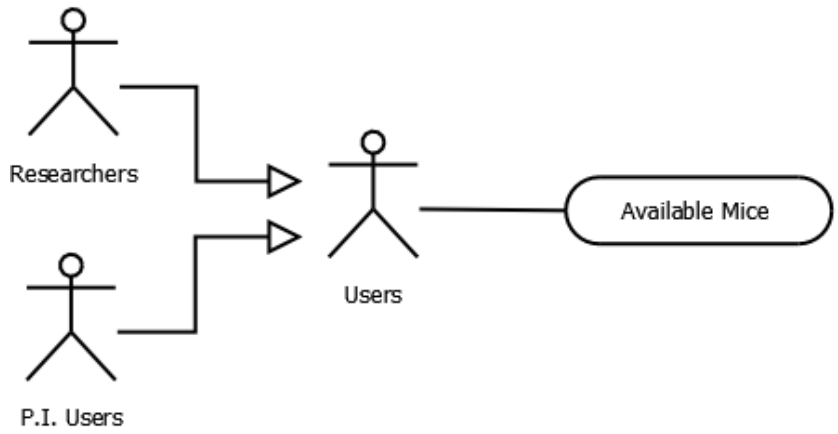

Figura 28: Diagramma UML relativo al caso d’uso dei topi disponibili

L’utente procede quindi a selezionare il pulsante nel pannello relativo all’operazione di impianto di uno xenopaziente che si trova all’interno dell’area di controllo. Anche in questo caso il pulsante riporta il numero di animali disponibili per l’operazione. Il sistema mostra il pulsante di redirect e la descrizione che riporta nuovamente il numero di topi disponibili per l’impianto. L’utente clicca sul pulsante di reindirizzamento alla pagina di impianto. Il sistema effettua il caricamento della pagina per l’operazione di impianto presente nel modulo degli xenopazienti. La condizione di terminazione è quindi data dalla visualizzazione da parte dell’utente della pagina per effettuare l’impianto. I flussi alternativi sono invariati rispetto ai casi d’uso precedenti.

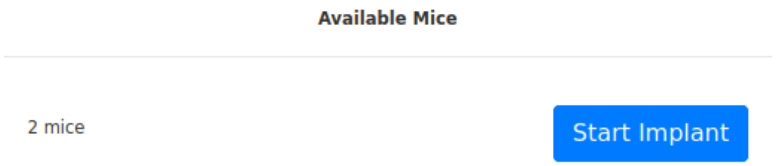

Figura 29: Dettaglio dell’area di visualizzazione

6.9 Mice Under Treatment

Il caso d’uso seguente è relativo al numero di topi sotto trattamento che sono elencati nel modulo degli Xenopazienti. La precondizione è nuovamente determinata dalla possibilità di accedere e visualizzare la pagina di riepilogo da parte dell’utente.

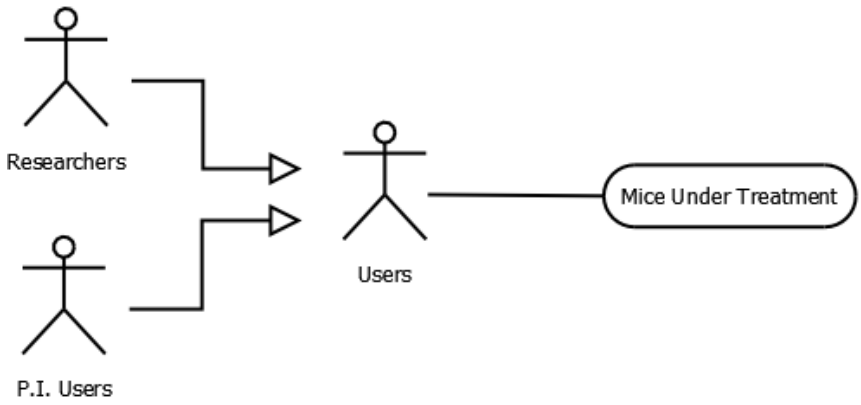

Figura 30: UML relativo agli Xenopazienti posti sotto trattamento

L'utente procede selezionando il pulsante nel pannello corrispondente a Mice Under Treatment presente nell'area di controllo. Il pulsante riporta il numero di animali sotto trattamento. Il sistema popola l'area di visualizzazione con una descrizione contenente il numero di animali sotto trattamento seguita dal pulsante di reindirizzamento alla pagina dedicata ai trattamenti nel modulo degli Xenopazienti. In seguito, l'utente clicca sul pulsante di redirect ed il sistema procederà a mostrare la pagina dedicata ai trattamenti. Quest'ultimo passaggio rappresenta la post condizione del caso in esame. Come per gli altri casi d'uso, rimangono invariati i flussi alternativi precedentemente individuati e trattati.

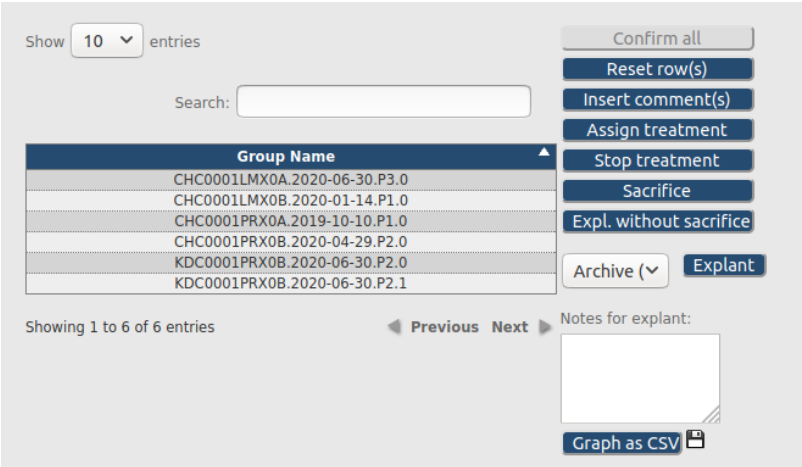

Figura 31: Dettaglio dei gruppi di topi sotto trattamento

6.10 Mice Explants

L'ultimo caso relativo al modulo degli Xenopazienti tratta gli animali pronti per la procedura di espianto. In questo caso d'uso l'utente procede selezionando il pulsante nel pannello corrispondente a Mice Explants situato all'interno dell'area di controllo.

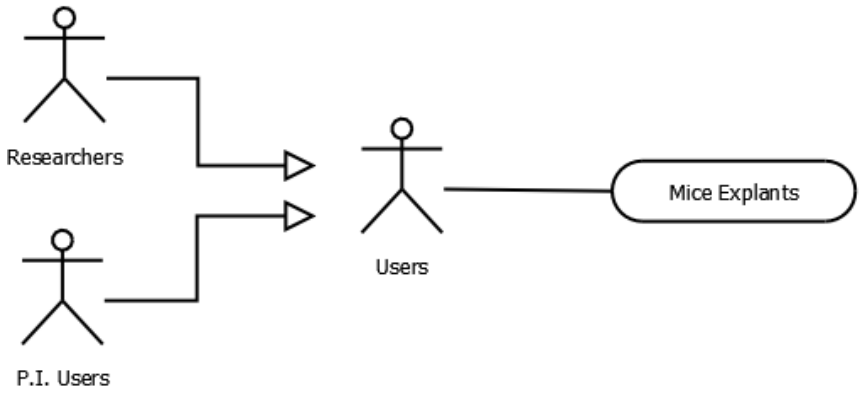

Figura 32: Diagramma UML relativo all'operazione di espianto

Il pulsante riporta il totale degli animali interessati dalla procedura di espianto, sia essa leggera o normale. Il sistema carica i dati nell'area di visualizzazione mostrando quanti topi sono pronti per l'espianto e quanti sono destinati all'espianto leggero, come mostrato in Figura 33. Inoltre, viene mostrato il pulsante per iniziare la procedura di espianto. L'utente clicca sul pulsante di reindirizzamento alla pagina per l'espianto. Il sistema procede col caricamento della pagina iniziale della procedura di espianto che è comune ad entrambe le operazioni (espianto normale e leggero). La condizione di terminazione è quindi la visualizzazione della pagina per l'espianto da parte dell'utente. Anche per questo caso d'uso, come visto nei precedenti, l'utente può interrompere l'attività in qualunque momento oltre a poter incorrere in eventuali errori da parte del sistema che impediscano, in parte o del tutto, l'utilizzo della piattaforma.

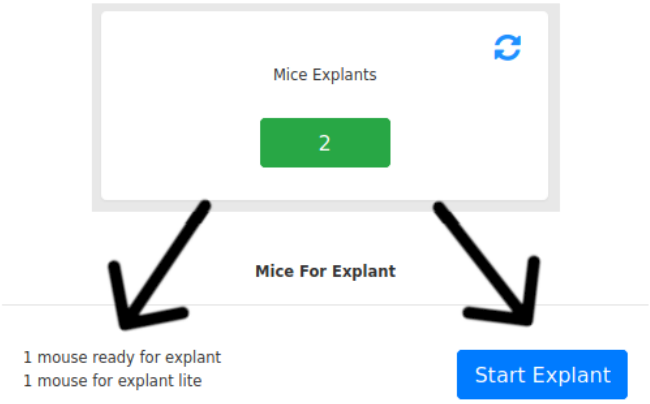

Figura 33: Dettaglio dell'area di visualizzazione per l'operazione di espianto

Completano l'elenco i casi d'uso appartenenti al terzo raggruppamento contraddistinto anch'esso dalla post condizione in comune, ovvero l'aggiornamento dell'aspetto grafico della pagina della dashboard. Queste operazioni non determinano quindi un reindirizzamento ad un modulo specifico ma vanno ad agire opportunamente sulla pagina di riepilogo stessa. Pertanto, i casi d'uso risultano più brevi e concisi rispetto ai precedenti date le minori interazioni che coinvolgono l'utente ed il sistema.

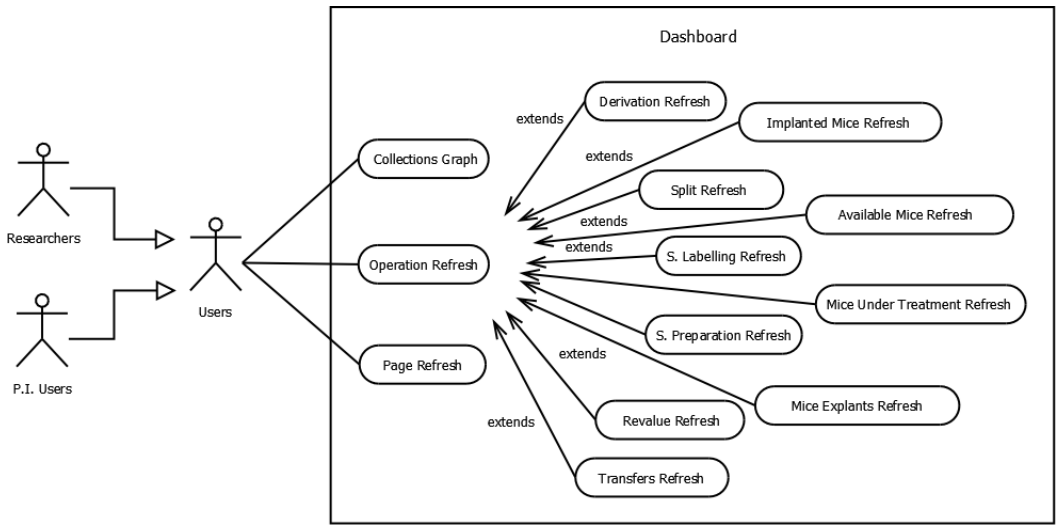

Figura 34: Diagramma UML casi d'uso relativi ai refresh ed al grafico

6.11 Collections Graph

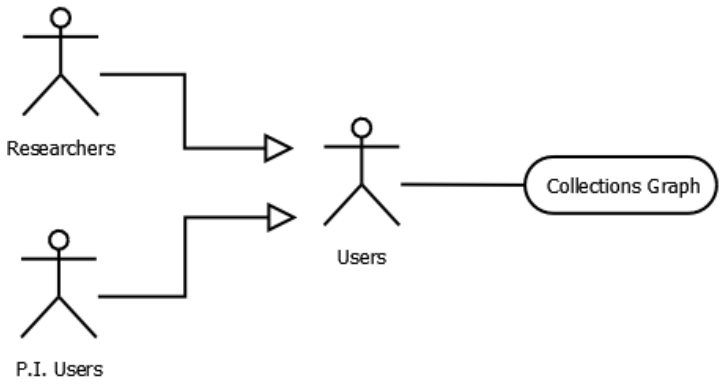

Figura 35: Rappresentazione UML relativa al grafico delle collezioni

Questo caso d'uso riguarda la visualizzazione del grafico contenente le collezioni presenti nel sistema. La condizione di partenza è determinata dall'accesso e dalla visualizzazione da parte dell'utente della pagina relativa alla dashboard. Successivamente, l'utente clicca sul pulsante all'interno del pannello relativo alle collezioni presente nell'area di controllo. In questo caso, il pulsante non presenta alcuna informazione preliminare per l'utente, viene mostrata una stringa indicante la funzione del pulsante. Il sistema procede mostrando il grafico aggiornato con legenda nell'area di visualizzazione. Questa rappresenta la normale terminazione del caso d'uso. Anche per questo scenario valgono i flussi alternativi evidenziati in precedenza.

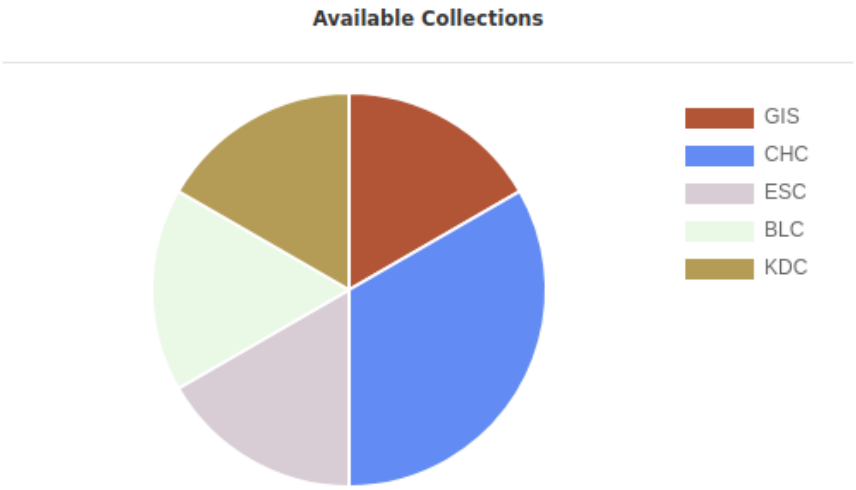

Figura 36: Esempio di grafico delle collezioni

6.12 Refresh delle operazioni

Come mostrato nel diagramma UML presente in Figura 37, questo caso d'uso va a descrivere le operazioni di refresh di ciascun pannello dell'area di controllo. È escluso dall'aggiornamento il pannello relativo al grafico delle collezioni in quanto non sono mostrati dati aggiuntivi all'interno dello stesso. Il caso d'uso è quindi comune a tutti i pannelli relativi alle operazioni della Biobanca e sugli Xenopazienti.

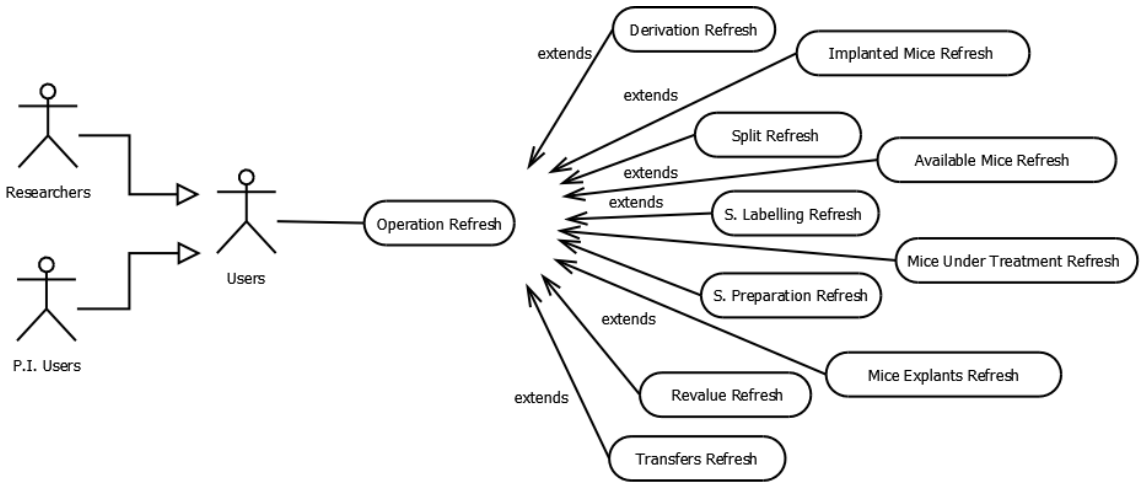

Figura 37: Diagramma UML relativo ai pulsanti di refresh

L'utente procede cliccando sull'icona di refresh presente nel pannello di controllo desiderato. Il sistema effettua l'aggiornamento del valore presente all'interno del pulsante. Viene prima impostato il valore temporaneo "Loading", al termine del caricamento l'utente vedrà l'indice aggiornato corrispondente al numero di aliquote o di xenopazienti. Il caso d'uso termina con la visualizzazione da parte dell'utente del nuovo valore all'interno del pulsante nel pannello interessato dall'aggiornamento. I flussi alternativi risultano invariati rispetto a quanto precedentemente descritto.

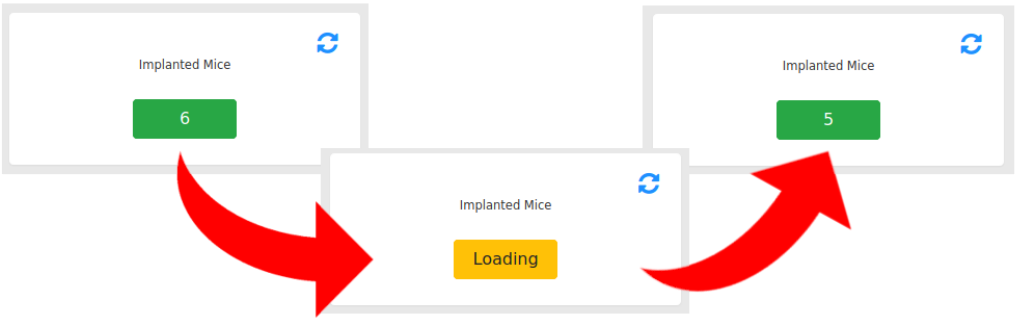

Figura 38: Esempio di step di aggiornamento di un pulsante

6.13 Page refresh

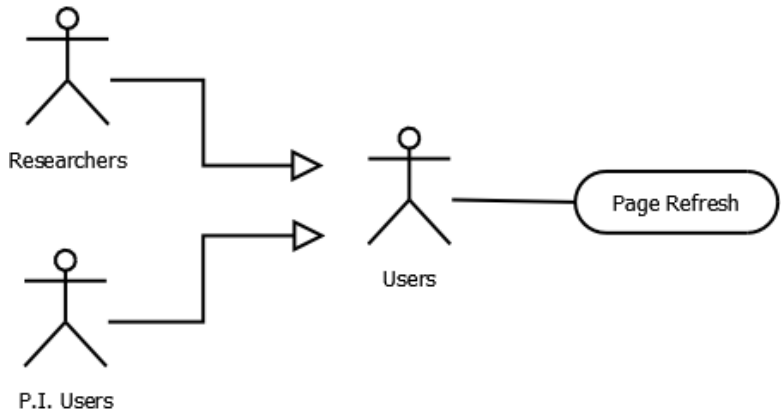

Figura 39: Diagramma UML legato al pulsante di refresh della dashboard

Questo caso d'uso prende in considerazione il pulsante di ricaricamento della pagina che effettua simultaneamente il caricamento dei valori all'interno di tutti i pulsanti dei pannelli di controllo. La precondizione, come in precedenza, è rappresentata dalla possibilità di accedere alla dashboard da parte dell'utente. Questo caso d'uso risulta molto semplice in quanto l'utente clicca sul pulsante di refresh della dashboard ed il sistema procederà a ricaricare i valori all'interno dei pannelli di controllo. Inoltre, il sistema azzererà l'area di visualizzazione impostando lo stato di default. Il caso si conclude con la visualizzazione della dashboard con i dati correttamente aggiornati. Le condizioni di terminazione anomale, mostrate nei flussi alternativi dei casi d'uso elaborati in precedenza, sono valide anche per il caso d'uso in analisi.

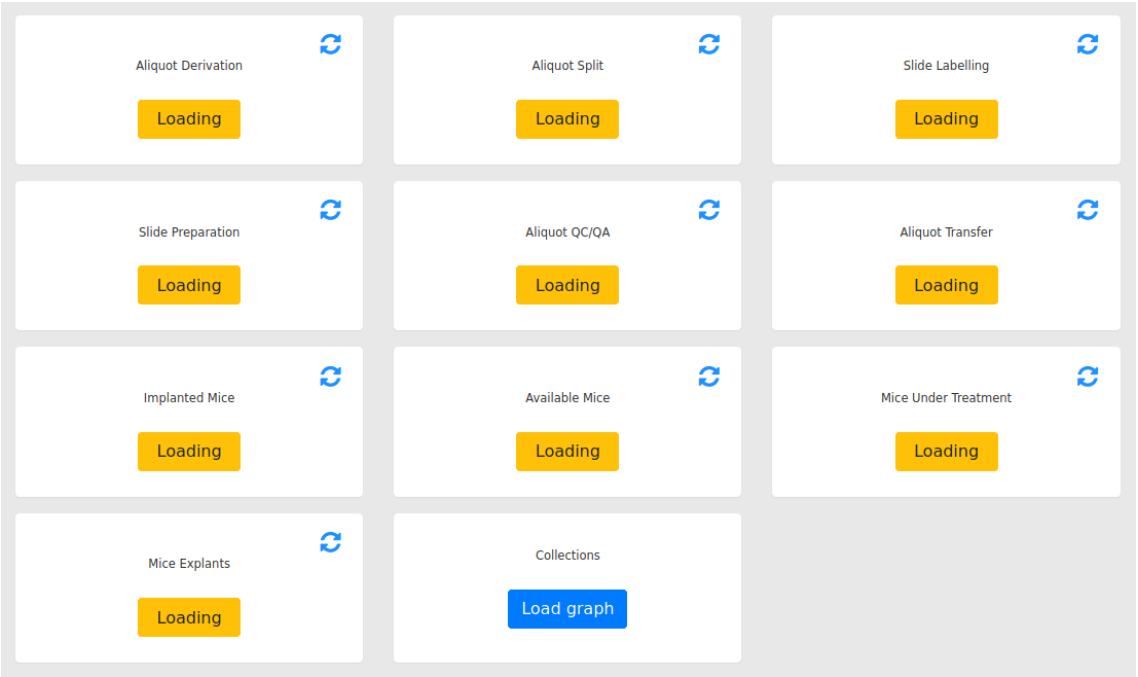

Figura 40: Esempio di refresh in corso dell'area di controllo

Queste descrizioni rappresentano i passaggi dei casi d'uso relativi alle operazioni che vengono eseguite nella pagina della dashboard. Le precondizioni sono le medesime per qualunque caso d'uso relativo alla pagina di riepilogo, essendo un'applicazione a pagina singola, viene eseguito un solo caricamento iniziale per rendere disponibile l'intero software. Inoltre, come discusso in precedenza nel capitolo, non tutte le tipologie di utenti del sistema possono visualizzare la dashboard. Pertanto, i prerequisiti di utilizzo dell'applicativo prevedono che l'utente appartenga ad una delle tipologie ammesse e che abbia effettuato l'accesso alla pagina. Il flusso principale, relativamente alla prima categoria di casi d'uso riguardanti la Biobanca, è caratterizzato dalla presenza di operazioni che possono essere eseguite sia manualmente che mediante robot. In generale, la normalità è rappresentata dalle operazioni eseguite in modalità manuale, questo vale per tutti i casi d'uso trattati. Pertanto, si è tenuta in considerazione la possibilità di avere queste estensioni del flusso qualora l'operazione preveda l'esecuzione tramite robot.

I casi d'uso si sviluppano tutti partendo dall'utente che procede ad una prima selezione dell'operazione da eseguire. Questa selezione viene operata fra i pulsanti contenuti nei pannelli dell'area di controllo. Ciascun pulsante contiene l'indicazione di quante aliquote o del numero di xenopazienti sono stati schedulati per l'operazione. Il sistema risponde

all'input ricevuto mostrando tutte le descrizioni ed i pulsanti relativi agli step previsti per la procedura scelta. Quest'ultimo passaggio avviene all'interno dell'area di visualizzazione il cui contenuto viene sostituito di volta in volta. La descrizione varia in base alle modalità di esecuzione previste ed al numero di step di cui è composta l'operazione scelta. Per le operazioni che prevedono l'esecuzione sia manuale che mediante robot, il valore della descrizione risulta essere il medesimo. Questo è dovuto alla procedura di schedulazione nella quale l'utente non può selezionare la modalità di esecuzione. Quest'ultima viene decisa in seguito nel modulo della Biobanca andando a selezionare la modalità di esecuzione dell'operazione tramite gli appositi pulsanti. La dashboard permette di effettuare questa selezione a monte, rimandando l'utente direttamente alla pagina di esecuzione.

Il caso d'uso prosegue portando l'attenzione dell'utente sui pulsanti inseriti nell'area di visualizzazione. Pertanto, lo scenario continua con la selezione da parte dell'utente dello step desiderato e della modalità di esecuzione, qualora ne venga presentata la scelta. In seguito, il sistema effettua il reindirizzamento dell'utente alla pagina iniziale dell'operazione scelta per poter concludere il lavoro. Al termine di ciascun caso d'uso, l'utente deve visualizzare la pagina prevista per l'esecuzione dell'operazione che intende concludere. Questo rappresenta la post condizione, ovvero la conclusione naturale del caso d'uso.

I casi d'uso appartenenti al terzo raggruppamento, relativi alle icone di aggiornamento, al pulsante di refresh generale ed al grafico delle collezioni, sono descritti mediante un flusso degli eventi più breve. Questo è dovuto al fatto che sono composti da una sola interazione fra l'utente ed il sistema, ovvero la pressione del pulsante o dell'icona corrispondente all'azione voluta dall'utente. Come visto in precedenza, la condizione di terminazione per questi casi d'uso risulta essere la visualizzazione della dashboard con i dati aggiornati.

Nei flussi alternativi vengono descritti i possibili scenari che non portano alla conclusione prevista dai casi d'uso. Si tratta di conclusioni dovute ad errori del sistema o ad altre cause non imputabili all'applicazione, quali ad esempio la chiusura della pagina da parte dell'utente. Qualora si presentino errori nel sistema a backend, la pagina è stata progettata per continuare a funzionare. Pertanto, la dashboard sarà fruibile anche in presenza di errori dovuti a dati errati o impropriamente modificati nel database. In questi casi, i pulsanti interessati dagli errori continuano a presentare la dicitura "Loading" anche al termine del caricamento. Questo perché non è possibile assegnare un valore in assenza di una risposta corretta da parte del backend. Qualora il sistema non sia raggiungibile la

pagina non verrà caricata oppure, nel caso in cui sia stata caricata in precedenza, non funzionerà correttamente.

Come si evince dalle elaborazioni precedenti, le condizioni di partenza sono le stesse per ogni caso d'uso elaborato. Si può giungere alla medesima conclusione per quanto riguarda i flussi alternativi previsti negli scenari d'errore sopra descritti. Questo accade in quanto la dashboard è un'applicazione a pagina singola. Pertanto, ogni suo componente deve sottostare alle stesse condizioni di utilizzo. Questo si ripercuote altresì sui possibili errori a cui il backend può andare incontro. La precisazione riguardo al backend è determinata dal fatto che non sono previsti errori da parte del frontend. Tutto ciò è possibile in quanto non vengono effettuate elaborazioni sui dati ricevuti, la pagina è strutturata per interrogare il server e visualizzare i dati ottenuti in risposta. Qualora il server non risponda o mandi dati errati, la pagina di riepilogo filtra questo tipo di risposte evitando di incorrere a sua volta in possibili errori. L'utente che utilizza il sistema noterà eventuali errori in quanto i pulsanti interessati conterranno la dicitura "Loading" e non un valore numerico.

Per la rappresentazione grafica dei casi d'uso si è deciso di impiegare tre diagrammi UML di dimensione minore contenenti i rispettivi raggruppamenti. Queste immagini, con annessa breve descrizione, sono state utilizzate a scopo introduttivo delle rispettive sezioni di analisi. Inoltre, sono state inserite immagini significative per la maggior parte dei casi d'uso trattati con lo scopo di evidenziare alcuni particolari legati nello specifico al caso d'uso oggetto di studio.

7 Conclusioni

L'obiettivo principale raggiunto dal progetto di tesi consiste nell'implementazione della pagina contenente il riepilogo delle attività in corso d'opera relative a ciascun utente. Per concretizzare tale obiettivo si è partiti con lo studio del funzionamento della piattaforma e delle entità in essa trattate. Questa analisi ha permesso di effettuare un censimento delle varie operazioni che vengono effettuate sui campioni. In seguito, si è proseguiti con lo studio dei cambiamenti di stato e dei passaggi fra le entità trattate. Analisi condotta mediante l'interpretazione del codice della piattaforma e le variazioni apportate ai dati nei database. Fra questi dati si è preso in considerazione anche il GenealogyID andando ad interpretarne la struttura e a verificare il valore rappresentato da ogni singolo segmento. In aggiunta al codice, sono state analizzate anche le tecnologie comprendenti i vari linguaggi di programmazione nonché i framework impiegati per lo sviluppo della piattaforma. Tutto ciò ha portato all'individuazione dei dati di maggior rilevanza per l'utente, informazioni che sono state inserite nella dashboard. Sulla base di questi dati si è passati alla fase di progettazione del software, andando ad individuare una struttura appropriata per la pagina al fine di mostrare correttamente tutte le informazioni. In questa fase si è tenuto conto altresì di possibili evoluzioni future della pagina di riepilogo, questo ha portato alla definizione di una struttura dinamica e facilmente estendibile. Il passaggio seguente ha comportato la realizzazione della dashboard e la successiva elaborazione dei casi d'uso. I test sono stati condotti seguendo i casi d'uso precedentemente stilati, i quali hanno portato all'implementazione di una serie di miglioramenti al software. Inoltre, questi aggiornamenti hanno introdotto nuove funzionalità che hanno portato all'elaborazione di nuovi casi d'uso legati ad esse.

In concomitanza dell'obiettivo primario, sono stati raggiunti anche alcuni obiettivi secondari. Fra questi abbiamo il censimento dei cambiamenti di stato delle entità, obiettivo necessario al completamento della dashboard in quanto punto di partenza per la creazione della struttura della pagina. L'analisi tecnica della piattaforma ha evidenziato la mancanza di una libreria o di un framework grafico per la gestione dei dispositivi mobile. Questa tipologia di dispositivi è diventata di uso comune in numerosi ambiti, fra i quali quello della ricerca scientifica. Durante la fase di progettazione della dashboard si è deciso di impiegare la libreria Bootstrap al fine di rendere la pagina responsive e in linea con le recenti modalità di sviluppo web. Dal punto di vista grafico, i moduli impiegati

nella piattaforma hanno alcuni aspetti in comune che potrebbero essere accorpati utilizzando un framework basato su componenti. Creando e riutilizzando uno stesso componente per la gestione di un aspetto in comune fra i vari moduli si andrebbe a ridurre il quantitativo di codice duplicato presente nei vari moduli. Oltre a ciò, nell'eventualità di dover effettuare modifiche, queste ultime verrebbero centralizzate tutte in un unico componente. Pertanto, possiamo includere fra gli obiettivi secondari l'individuazione di potenziali miglioramenti applicabili all'intera piattaforma. Per lo sviluppo della dashboard si è deciso di non impiegare framework particolari oltre a quanto già disponibile nel sistema. Qualora in futuro venga adottata una strategia che prevede l'impiego di un framework per il frontend, si dovrà creare un componente ad hoc per la dashboard in quanto non presenta aspetti in comune con il resto del software. Oltre a ciò, essendo la piattaforma non del tutto completa, si è utilizzato l'analisi funzionale come strumento di test del software. Sono state individuate e segnalate agli sviluppatori alcune procedure che presentavano comportamenti errati, mentre altre operazioni non restituivano all'utente un feedback corretto sulla mancanza di parametri o dati necessari per l'esecuzione delle stesse.

Come è stato rimarcato in precedenza, per la realizzazione della dashboard è stato ritenuto opportuno effettuare un'analisi approfondita della struttura e delle funzioni del GenealogyID all'interno della piattaforma. Questo perché, attraverso la conoscenza del significato di ciascun valore presente nei segmenti di cui il codice è composto, è possibile individuare immediatamente la tipologia di entità trattata. Oltre a ciò, il codice viene impiegato per tracciare la storia di un determinato campione ed è il mezzo con cui lo si identifica nei vari moduli della piattaforma in cui viene utilizzato. Queste caratteristiche rendono fondamentale la comprensione del GenealogyID per futuri sviluppatori nonché per i futuri utilizzatori della piattaforma. L'analisi effettuata nei capitoli precedenti può quindi essere utile per nuovi progetti di studio così come per lo sviluppo in futuro di nuove componenti andando ad offrire una descrizione concisa e dettagliata dei significati assunti dai vari segmenti.

Durante una prima fase di analisi della piattaforma si è deciso di effettuare una breve comparazione con altri sistemi simili al fine di individuare eventuali miglioramenti adottabili anche per il LAS. Le principali funzioni aggiuntive, introdotte nei sistemi competitor, riguardano aspetti di natura gestionale piuttosto che commerciale, pertanto più orientati al mondo business ed alle aziende in generale. Lo scopo principale del LAS è quello di sostenere la ricerca nei laboratori e di mantenere un ambito di utilizzo a carattere prettamente scientifico, pertanto queste funzionalità non avrebbero motivo di essere integrate nel sistema attuale.

Un aspetto di maggior importanza, che necessita di essere rivisto, viene riscontrato nella staticità grafica della piattaforma. Allo stato attuale, tecnicamente è possibile fruire del software su qualsiasi dispositivo. Dal punto di vista pratico questo non è del tutto possibile in quanto le pagine non si adattano agli schermi di dimensioni ridotte andando a ridurre quasi del tutto a zero la possibilità di fruizione del sistema. Sotto questo punto di vista, la dashboard si presenta come un'applicazione a pagina singola correttamente predisposta per essere compatibile con la maggior parte dei device ad oggi in commercio. Il target preso in considerazione per lo sviluppo dell'interfaccia grafica è rappresentato dai tablet, oltre chiaramente a tutti i terminali fissi e laptop. Questo perché il tablet risulta essere un dispositivo mobile il cui schermo può raggiungere una dimensione adatta all'impiego dell'attuale piattaforma. L'utilizzo su smartphone della dashboard è altrettanto possibile, lo stesso non si può dire per il resto del sistema per i motivi sopra citati.

Il backend è stato sviluppato seguendo il più possibile un approccio di tipo REST. Pertanto, sono state implementate apposite funzioni, le API nei moduli della Biobanca e degli Xenopazienti, per reperire le informazioni di interesse per l'utente direttamente dal database. La struttura della dashboard si presta particolarmente bene all'impiego di chiamate REST in quanto ogni pannello dell'area di controllo è associato ad una determinata operazione. Quindi, ciascun pannello interagisce con una sola risorsa presente nel backend, la quale è raggiungibile mediante apposito URL. Le interazioni col database sono limitate alla sola lettura dei dati in esso presenti. Questo perché la pagina di riepilogo è stata strutturata per mostrare una serie di informazioni utili all'utente per poter tornare velocemente a lavorare su una o più operazioni lasciate in sospeso.

7.1 Sviluppi futuri

La dashboard è stata progettata per avere una struttura facilmente estensibile. Questo concetto è stato introdotto in precedenza nel capitolo dedicato allo sviluppo della pagina di riepilogo. La struttura dell'area di controllo è stata pensata per permettere di inserire un numero potenzialmente illimitato di pannelli per la gestione di specifiche funzionalità. Ciascuno di essi si disporrà automaticamente di seguito ai pannelli già presenti nella pagina andando ad accrescere verticalmente la dimensione della pagina. A questo punto, sarà necessario operare una decisione di carattere estetico, che consiste nel decidere se mantenere uguale la lunghezza delle due sezioni oppure se rendere statica la lunghezza

dell'area di visualizzazione, andando quindi ad impostare un valore fisso per l'altezza. Questa considerazione riguarda esclusivamente l'aspetto grafico della pagina la cui importanza è chiaramente minore rispetto all'aspetto funzionale, ciò non implica che debba essere trascurato in quanto i dettagli grafici tendono ad incrementare l'usabilità dei software oltre a renderli più accattivanti.

Dal punto di vista delle funzionalità, le principali attività integrabili nella dashboard riguardano le linee cellulari e tutte le operazioni che vengono eseguite nei moduli riservati agli esperimenti. In merito alle linee cellulari è possibile mostrare quante piastre sono presenti nella Biobanca e quante di esse sono state schedulate per essere archiviate. Le colture cellulari eliminate non vengono più utilizzate nel sistema, pertanto quest'informazione non comporta nessun vantaggio per l'utente. Mentre le linee programmate per l'espansione vengono processate nel momento in cui l'utente decide di eseguire l'operazione. In questo caso, non essendo presenti punti di interruzione dell'operazione, non è possibile inserire questa tipologia di dato nella dashboard. Come precedentemente descritto, la piattaforma permette altresì di tenere traccia di numerose tipologie di esperimenti condotti in laboratorio. Il software gestisce gli esperimenti con le stesse modalità utilizzate per le operazioni condotte sulle aliquote. Questo porta ad avere una fase iniziale di pianificazione seguita da una fase di esecuzione dell'esperimento, quest'ultima potrà essere eventualmente divisa in più step a seconda della tipologia di esperimento condotto. È quindi possibile estendere ulteriormente la dashboard andando ad inserire nell'area di controllo un pannello ad hoc per ciascuna tipologia di esperimento trattato.

Altre funzionalità altrettanto valide per l'inserimento nella dashboard, potrebbero essere legate a nuove tipologie di statistiche sui campioni effettuate mediante apposite interrogazioni col modulo MDDM. Per il reperimento di questi dati sarà opportuno creare una o più query ad hoc. Le query saranno consultabili con le stesse modalità impiegate per la raccolta dei dati nel diagramma delle collezioni, ovvero mediante le API offerte dal MDDM. Queste interrogazioni dovranno essere salvate nel database e disponibili fin dal primo avvio del software, quindi dovranno essere inserite nei file di configurazione iniziale del database.

Analizzando la dashboard sotto un punto di vista diverso, sono state individuate due tipologie di utenti che hanno accesso alla pagina di riepilogo, i ricercatori ed i Principal Investigator (P.I.). Siccome i P.I. sono una tipologia di utenti con maggiori privilegi all'interno della piattaforma, potrebbe essere utile a questa categoria di utenti avere una sezione della dashboard dedicata ad operazioni di controllo del lavoro condotto dai ricercatori appartenenti ai loro gruppi. Si potrebbe quindi inserire una sezione nuova o

alcuni pannelli di controllo dedicati a questa tipologia d'utenza. La prima scelta comporterebbe una nuova analisi della struttura della dashboard per identificare il modo più corretto per mostrare agli utenti P.I. la nuova sezione di supervisione. Inoltre, si dovrà individuare una modalità intuitiva di passaggio fra la sezione di supervisione e l'area di controllo classica. L'inserimento di pannelli di controllo dedicati comporta uno sforzo implementativo minore in quanto è sufficiente decidere in quale posizione dell'area di controllo verranno mostrati. L'inserimento di questi nuovi pannelli avverrà programmaticamente e con le stesse modalità utilizzate per i pannelli già presenti nella dashboard.

Tutte queste nuove funzionalità dovranno tenere in considerazione la visibilità che gli utenti e i loro gruppi hanno rispetto ai campioni conservati nella Biobanca. Sarà quindi necessario garantire che ciascun gruppo abbia accesso esclusivamente ai dati di propria competenza, come è stato implementato nella dashboard attuale. Inoltre, qualora venga deciso di integrare funzionalità specifiche per il tipo di utenza, sarà necessario inserire un ulteriore controllo per impedire che gli utenti di tipo ricercatore abbiano accesso a funzionalità dedicate agli utenti di tipo Principal Investigator.

8 Bibliografia

[1] Grand,A., Geda,E., Mignone,A. et al. One tool to find them all: a case of data integration and querying in a distributed LIMS platform. Database (2019) Vol. 2019: article ID baz004; doi:10.1093/database/baz004

[2] Bika Open Source LIMS project. Tratto il 16 febbraio 2020, 10:05 da https://www.bikalims.org

[3] Open-LIMS - The Open-Source Laboratory Information Management System. Tratto il 16 febbraio 2020, 10:05 da http://www.open-lims.org

[4] Plone. Tratto il 16 febbraio 2020, 10:05 da https://plone.org

[5] PHP. Tratto il 16 febbraio 2020, 10:05 da https://www.php.net

[6] PostgreSQL: The World's Most Advanced Open Source Relational Database. Tratto il 16 febbraio 2020, 10:05 da https://www.postgresql.org

[7] Docker. Tratto il 22 marzo 2020, 11:14 da https://www.docker.com

[8] Apache HTTP Server Project. Tratto il 22 marzo 2020, 11:14 da https://httpd.apache.org

[9] Java. Tratto il 3 maggio 2020, 09:30 da https://www.java.com

[10] Python. Tratto il 3 maggio 2020, 09:30 da https://www.python.org

[11] Django. The web framework for perfectionists with deadlines. Tratto il 3 maggio 2020, 09:30 da https://www.djangoproject.com

[12] Pyramid. The Start Small, Finish Big Stay Finished Framework. Tratto il 3 maggio 2020, 09:30 da https://trypyramid.com

[13] Node.js. Tratto il 3 maggio 2020, 09:30 da https://nodejs.org

[14] React. Una libreria JavaScript per creare interfacce utente. Tratto il 7 giugno 2020, 10:10 da https://it.reactjs.org

[15] Angular. One framework. Mobile & desktop. Tratto il 7 giugno 2020, 10:10 da https://angular.io

[16] TypeScript. JavaScript that scales. Tratto il 7 giugno 2020, 10:10 da https://www.typescriptlang.org

[17] Vue.js. The Progressive JavaScript Framework. Tratto il 7 giugno 2020, 10:10 da https://vuejs.org

[18] jQuery. Write less do more. Tratto il 7 giugno 2020, 10:10 da https://jquery.com

[19] jQuery User Interface. Tratto il 21 giugno 2020, 11:06 da https://jqueryui.com

[20] JSON. Tratto il 21 giugno 2020, 11:06 da https://www.json.org/json-en.html

[21] MySQL. The world's most popular open source database. Tratto il 21 giugno 2020, 11:06 da https://www.mysql.com

[22] Neo4j. Tratto il 21 giugno 2020, 11:06 da https://neo4j.com

[23] MongoDB. The database for modern applications. Tratto il 21 giugno 2020, 11:06 da https://www.mongodb.com

[24] Fiori, A., Grand, A., Alberto, P., Geda, E., Brundu, F. G., Schioppa, D., & Bertotti, A. (2014). A Case Study of a Laboratory Information System Developed at the Institute for Cancer Research at Candiolo. Laboratory Management Information Systems: Current Requirements and Future Perspectives: Current Requirements and Future Perspectives, 252.

[25] Build fast, responsive sites with Bootstrap. Tratto il 07 luglio 2020, 18:06 da https://getbootstrap.com

[26] Chart.js. Simple yet flexible JavaScript charting for designers & developers. Tratto il 07 luglio 2020, 18:06 da https://www.chartjs.org