Indice generale

1 Introduzione

Contentuto del capitolo

1.1 Premessa

Obiettivo di questa tesi è la descrizione del processo di progettazione e realizzazione di una applicazione web per la gestione di dati di espressione genica derivanti da esperimenti di DNA microarray. In particolare, ci riferiamo ad una applicazione dotata di funzionalità di indicizzazione che consentano la ricerca attraverso i dati degli esperimenti archiviati. La problematica della ricerca costituisce il principale problema affrontato.

Il nostro punto di partenza è stato il lavoro di tesi di Giuseppe Battinelli, matr. 41/3024, che aveva progettato e realizzato una applicazione web per l'archiviazione di dati di espressione genica. Partendo da quanto progettato da Giuseppe Battinelli, è stato nostro compito creare una applicazione completamente nuova, dotata di più ampie funzionalità, che consentisse di effettuare ricerche nei dati degli esperimenti. Si rimanda al lavoro di Battinelli per una panoramica sulla ricerca genetica e la sperimentazione con i DNA microarray.

Gli esperimenti di DNA microarray sono molto complessi e assumono le forme più varie. Noi abbiamo studiato il modello dei dati prodotto da Battinelli, lo abbiamo ulteriormente validato con la collaborazione dei ricercatori dell'Istituto di Scienze dell'Alimentazione di Avellino, e abbiamo apportato ogni modifica che abbiamo ritenuto opportuna fino a produrre un modello nuovo.

Il modello prodotto, come il predecessore, è compatibile con le specifiche MIAME 1.1, che costituiscono lo standard “de facto” per la descrizione di esperimenti di espressione genica. Ciò ha reso l'applicazione che ne è derivata capace di memorizzare, in maniera chiara e completa, i dati prodotti con le tecniche più disparate, salvaguardando il principio della riproducibilità dell'esperimento.

La sperimentazione è sempre frutto delle preziose conoscenze dei ricercatori, per questo abbiamo concentrato le nostre attenzioni sulla sicurezza dei dati, introducendo diversi livelli di autorizzazione in grado di far fronte ai più semplici tentativi di attacco. Non ci azzardiamo nemmeno a dire di aver prodotto un sistema sicuro, ma speriamo di aver posto la forzatura dell'applicazione al di fuori delle possibilità di un utente mediamente esperto.

Speriamo di aver proposto un supporto alla condivisione dei dati di espressione genica, piuttosto che alla mera archiviazione. Questo lavoro, infatti, va oltre lo storage e si propone di offrire al mondo della ricerca uno strumento di sharing dei dati; uno strumento capace di raccogliere dati e, tramite l'inclusione di un motore di ricerca, di metterli a disposizione del vasto pubblico dei ricercatori per indagini, approfondimenti e confronti.

1.2 Lo stato dell'arte nella descrizione degli esperimenti, MIAME

La biologia molecolare, la genetica e la biochimica producono oggi una enormità di informazioni, una quantità di dati tale da non poter essere studiata senza utilizzare un supporto informatico. Il ruolo dell'informatica nell'elaborazione e nell'interpretazione degli output della ricerca è divenuto via via più importante, fino a divenire indispensabile, dando luogo alla nascita di una nuova disciplina: la bioinformatica.

La neonata bioinformatica si occupa di fornire modelli statistici validi per l'interpretazione dei dati, al fine di identificare tendenze e leggi numeriche; genera nuovi modelli e strumenti matematici per l'analisi di sequenze di DNA, RNA e proteine al fine di acquisire conoscenze relative alla frequenza di sequenze rilevanti, alla loro evoluzione ed eventuale funzione; organizza le conoscenze acquisite a livello globale su genoma e proteoma in basi di dati al fine di rendere tali dati accessibili a tutti

Se restringiamo la nostra analisi al mondo dei DNA microarray, il primo problema che oggi si cerca di affrontare è quello dell'archiviazione dei dati in modo chiaro, efficace e completo, in modo che questi siano disponibili alla comunità scientifica per i futuri studi. La dimensione dei dati in questo caso assume un ruolo fondamentale: normalmente studi che contengono 100 analisi per paziente per 1000 pazienti possono essere considerati vasti studi clinici. Uno studio microarray di media vastità, invece, comprende diverse migliaia di dati per campione su centinaia di campioni diversi.

Ai problemi legati all'enorme mole di dati prodotti si aggiunge il problema della diversità delle forme in cui si presentano. La mancanza di standardizzazione negli array presenta un problema interoperativo nella bioinformatica, che però non deve impedire lo scambio dei dati ottenuti con tale tecnica. Diversi progetti opensource si prefiggono di facilitare l'interscambio di dati ottenuti da microarray: tra di essi il "Minimum Information About a Microarray Experiment" (MIAME), uno standard per la descrizione di esperimenti, è stato adottato da molti giornali scientifici per l'accettazione dei lavori da pubblicare.

MIAME, un prodotto di MGED (Microarray Gene Expression Data), si propone di specificare tutte le informazioni necessarie ad interpretare i risultati di un esperimento in modo non ambiguo, garantendo la riproducibilità dello stesso. Sebbene lo standard specifichi l'insieme di contenuti che è necessario raccogliere per produrre report di esperimenti, esso non indica il formato in cui i dati debbono essere presentati. I dati, infatti, si presentano in una varietà di formati diversi, dettati dai produttori di array e dai software usati per l'analisi.

Lo standard prevede che le informazioni minime per la descrizione di un esperimento siano divise in sei sezioni:

1. design dell'esperimento,
2. design dell'array,
3. i campioni utilizzati,
4. le ibridazioni,
5. le misure relative all'acquisizione delle immagini,
6. il tipo di normalizzazione e le analisi statistiche.

Evidentemente queste sei sezioni pongono l'accento su tre aspetti diversi: il design dell'esperimento, il design dell'array utilizzato nell'esperimento e le tecniche di analisi dei dati. Illustreremo ognuno di essi nel seguito senza soffermarci sui dettagli, perché questi saranno approfonditi nella fase di modelling dei dati.

1.2.1 L'esperimento

Tutti gli esperimenti di DNA microarray cominciano da un campione biologico, ad esempio da una biopsia o da una coltura cellulare. Questo campione viene estratto l'RNA, poi etichettato ed ibridato su un chip. Generalmente si lavora su più di un campione, per confrontare una coppia di colture, un campione trattato e uno di riferimento, o anche una serie di campioni prelevati da un time-course.

La prima parte di MIAME chiede semplicemente di descrivere il design dell'esperimento, l'origine dei campioni e le relazioni tra loro. In aggiunta lo standard chiede che i protocolli utilizzati per estrarre gli RNA, per etichettarli e per generare le ibridazioni, siano descritti insieme ai valori dei parametri utilizzati.

Le informazioni minime in questa sezione includono la tipologia dell'esperimento (ad esempio confronto normal-versus-diseased, time course, dose response, e così via) e le variabili sperimentali, inclusi i parametri o le condizioni testate (ad esempio time, dose, genetic variation o risposta ad un trattamento). Questa sezione deve anche includere gli indicatori di qualità, come l'utilizzo di replicati e i passi di ogni quality check. Queste notizie, espresse mediante l'uso di un vocabolario controllato, consentono future indagini e ulteriori analisi più di ogni altra descrizione testuale.

Infine bisogna specificare le relazioni sperimentali tra gli array e i campioni, cioè quale array e quale campione è stato utilizzato in ogni ibridazione. Ognuno di essi deve essere dotato di un identificativo univoco in modo che sia possibile far riferimento alle informazioni contenute nelle altre sezioni.

1.2.2 L'analisi dei dati

Preparato l'array, questo viene scannerizzato, e l'immagine prodotta viene utilizzata per generare una serie di numeri, ognuno dei quali rappresenta l'intensità di una particolare punto sul chip. Tipicamente una ulteriore processazione dei dati è necessaria a trasformare questi valori in qualcosa che rappresenti l'abbondanza di ogni mRNA nell'estratto originale. Differenze nel labelling e nell'ibridazione, così come differenze realizzative degli array, possono influenzare l'intensità degli spot.

Per rendere confrontabili i chip provenienti da produttori differenti, viene applicata una normalizzazione che porta i risultati ottenuti da ognuno in una riga. Effettuata la normalizzazione, i dati possono finalmente essere analizzati e interpretati; un set di geni o profili di espressione può essere prodotto.

MIAME chiede che sia reso disponibile un sito web o un database online che dia accesso ai dati grezzi, l'insieme dei livelli di espressione generati da ogni array, e ai risultati della loro manipolazione statistica. Lo standard chiede anche che questi dati siano resi disponibili in una forma che consenta ulteriori analisi; non chiede che siano rese disponibili le immagini originali, ma che gli autori mettano a disposizione i dati derivanti dalla scansione di queste, insieme ad una descrizione dettagliata dei valori dei parametri utilizzati per ottenere i livelli di espressione, inclusi i software utilizzati per l'analisi e i valori di ogni parametro che essi impieghino.

In aggiunta ai dati grezzi, MIAME chiede che ogni esperimento sia accompagnato dai dati processati (normalizzati) e dai risultati che ne derivano, ancora una volta, con una dettagliata descrizione di come sono stati generati questi stessi dati. La forma dei dati normalizzati dipende dall'analisi statistica cui sono stati sottoposti, ma l'importante è che questi siano forniti in forma tabellare, su un database online, in modo che siano possibili ulteriori analisi.

1.2.3 L'array

Chiaramente conoscere esattamente cosa è legato ad ogni spot (probe) presente sull'array è un prerequisito per l'interpretazione di un esperimento di microarray. Per questo motivo lo standard MIAME chiede che sia descritto in dettaglio ogni aspetto riguardante il design dell'arrray. In generale è sufficiente specificare il produttore dell'array e il nome del particolare modello, in modo che il lettore possa risalire al data sheet distribuito dal produttore stesso.

Bisogna fornire una definizione sistematica dell'array utilizzato, inclusi i geni rappresentati e la loro distribuzione fisica sull'array. Questo può avvenire per mezzo del manifest, o tabella delle annotazioni, distribuito dal produttore. Nel contesto di un base di dati per l'archiviazione degli esperimenti, ogni array dovrebbe essere immesso una sola volta e poi riferito da tutti gli utilizzatori.

Nel corso della stesura di questo lavoro abbiamo spesso utilizzato la parola platform, piattaforma, in luogo del vocabolo array. Abbiamo voluto così riferirci agli array come piattaforme tecnologiche, piuttosto che come microchip. Il lettore deve comunque sapere che array, chip e piattaforma, per noi hanno indicato la stessa entità.

1.3 Lo stato dell'arte nello sviluppo di applicazioni, i framework

Nel corso della breve storia dell'informatica lo sviluppo di software è rapidamente divenuto una attività ripetitiva. Sempre più spesso, a valle del processo di produzione, gli sviluppatori si sono resi conto di aver perso tempo a “reinventare la ruota” invece che a studiare soluzioni ai problemi concreti. Così si è diffusa la percezione che fosse necessario trovare soluzioni che consentissero di spendere una quantità maggiore di tempo nell'analisi dei requisiti e nella costruzione di un business model solido, piuttosto che nel debug dei flussi di controllo.

La risposta attuale a questo problema sta nell'utilizzo dei cosiddetti framework, strutture di supporto su cui progettare e organizzare il software. Un framework, che nella maggior parte dei casi è sviluppato in un linguaggio object-oriented, consta sempre di una serie di librerie di codice riutilizzabile e da un insieme classi astratte e relazioni tra le stesse, su cui costruire il proprio software. Esso solleva lo sviluppatore dalla progettazione di tutte quelle funzionalità secondarie, indipendenti dallo scopo per cui si sviluppa una particolare applicazione, che comunque competono alla costruzione del prodotto software.

E' importante sottolineare la differenza sottile, ma molto importante tra una libreria di software e un framework. Una library contiene una serie di funzionalità che una qualunque applicazione può invocare. Un framework, d'altro canto, fornisce componenti generici in grado di cooperare tra loro, che una qualunque applicazione può estendere per fornire un particolare set di funzionalità. I punti in cui un framework può essere esteso sono detti extension point. Per questo motivo spesso ci si riferisce ai framework come librerie “upside-down”.

In generale un framework presenta le seguenti carateristiche:

- include molte classi e/o componenti ognuno dei quali può fornire una astrazione di un particolare concetto;
- definisce come queste astrazioni coopereranno nel funzionamento dell'applicazione;
- è composto di componenti riutilizzabili;
- organizza il pattern progettuale ad un elevato livello di astrazione.

In genere i vantaggi dell'utilizzo di un framework vanno ben oltre gli svantaggi, anzi si può affermare che quanto più il progetto sia di grosse dimensioni tanto più l'utilizzo di un framework è altamente consigliabile. In effetti, poiché ogni framework impone l'utilizzo di una propria architettura e un particolare modo di avvicinarsi ai problemi, il suo utilizzo presuppone uno step ulteriore: lo studio del framework stesso. Così è poco conveniente studiare l'architettura di un framework per realizzare un applicativo di piccole dimensioni.

Di seguito vengono schematicamente riassunti alcuni dei principali vantaggi che si ottengono nell'adozione di un framework nello sviluppo di applicazioni.

- un framework è fondato su un disegno architetturale valido, in quanto il suo codice è scritto in base alle best-practises della tecnologia in uso. Ciò conferisce al proprio progetto fondamenta solide dalle quali partire;
- lo sviluppatore deve implementare esclusivamente la logica applicativa potendo risparmiare le energie e il tempo necessari alla scrittura di componenti infrastrutturali;
- un framework semplifica lo sviluppo applicativo perché fornisce tutta una serie di componenti che risolvono la gran parte dei compiti comuni a tutte le applicazioni (controllo del flusso, logging, gestione messaggi di errore, user interface, internazionalizzazione, validazione dei dati, etc..).

Ogni framework porta con se il proprio pattern progettuale e un insieme di funzionalità, per questo è importante adottare quello giusto, quello che si adatta ai requisiti dell'applicativo da sviluppare. Scegliere il framework sbagliato può rallentare, invece che accelerare, il processo di produzione del software. E' importante scegliere framework generici, che non impongano l'utilizzo di strumenti proprietari e che non siano legati ad una particolare infrastruttura hardware/software. Inoltre è importante utilizzare un framework che consenta il riuso della logica applicativa sviluppata anche al di fuori della struttura del framework stesso.

L'area in cui è più sentita la necessità di un framework è proprio quella delle applicazioni web. Qui l'offerta open source è tanto ampia e variegata da rendere difficile perfino la catalogazione dei prodotti disponibili. Nell'impossibilità di confrontare tutti i framework prodotti nel mondo open source, è necessario accontentarsi di una comparazione basata sul numero di utilizzatori nel mondo e sulla maggiore o minore vitalità delle community che li supportano.

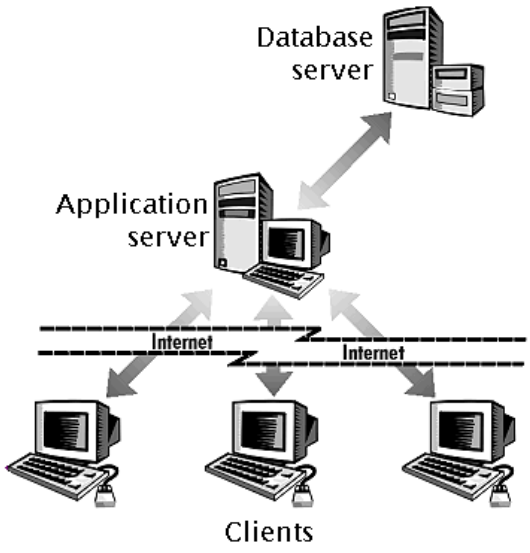

Figura 1.1.: Tipica architettura di una applicazione web.

I framework più in voga si basano sul modello Model View Controller (MVC), offrendo ognuno una variante più o meno originale del modello. Offriremo nel seguito una panoramica più dettagliata del modello. Per ora ci basti sapere che ci riferiamo allo sviluppo di applicazioni distribuite in una modalità che somiglia molto a quanto previsto dall'architettura three tier, che prevede la suddivisione del sistema in tre diversi strati dedicati rispettivamente alla intefaccia utente, alla logica funzionale (business logic) e alla gestione dei dati persistenti.

Nel modello three tier, come nel modello MVC, gli strati comunicano tra loro secondo il paradigma client-server, essendo l'interfaccia utente client della logica funzionale, a sua volta client della gestione dati. Una soluzione del genere prevede, nel caso web, un insieme di macchine client che accedono all'interfaccia utente per mezzo di un browser, un application server su cui sono in esecuzione la business logic e la logica di controllo, e un database server per la gestione dei dati. La figura 1.1 offre una visione schematica di una simile architettura.

2 Modelling dei dati

Contenuto del capitolo

2.1 Analisi dei requisiti

Lo scopo generale dell'analisi dei requisiti è stabilire “che cosa” il sistema in questione debba saper fare, mentre le decisioni sul “come” sono rimandate alle successive fasi di progettazione. Bisogna, quindi, individuare con la massima precisione possibile quali sono i problemi che si vogliono affrontare utilizzando il sistema. In pratica, si vogliono definire funzioni, vincoli, prestazioni, interfacce e qualsiasi altra caratteristica che il sistema dovrà possedere per soddisfare le necessità del committente.

Siamo partiti dai requisiti individuati nella tesi dello studente che ci ha preceduti (Giuseppe Battinelli 041/003023) integrati da quanto di nuovo è emerso dalla comunicazione con i committenti. Abbiamo ampliato i requisiti con i problemi posti dal relatore e, dopo una serie di incontri/interviste con i futuri utenti del servizio, ricercatori nel campo della biologia molecolare, abbiamo focalizzato la nostra attenzione su quanto sintetizzato a seguire.

Si vuole realizzare una applicazione web che sia di supporto al lavoro dei ricercatori che utilizzano la tecnica DNA Microarray. L'applicazione deve essere in grado di archiviare i dati derivanti dalla sperimentazione di modo che questi siano disponibili per la ricerca. I dati devono essere archiviati insieme a tutte quelle informazioni che sono necessarie ad una interpretazione non ambigua dei risultati dell'esperimento e alla potenziale riproduzione dello stesso. In particolare si vuole che l'applicazione segua le specifiche imposte dallo standard MIAME 1.1.

L'applicazione si rivolge essenzialmente a due tipologie di utenti: gli amministratori e gli utenti registrati; essendo i primi un sottoinsieme dei secondi. Chiunque visiti anonimamente le pagine dell'applicazione deve avere la possibilità di visualizzare una interfaccia dalla quale effettuare una richiesta di registrazione, immettendo i

propri dati personali e scegliendo un nome utente e una password. Chiunque abbia fatto richiesta di registrazione deve essere attivato da uno degli amministratori, prima di poter accedere al sistema come utente registrato. Gli utenti registrati devono essere dotati di una interfaccia user-friendly per l'inserimento di nuovi esperimenti e per la visualizzazione, la modifica o l'eliminazione dei dati inseriti. Gli amministratori, in qualità di utenti registrati dotati di privilegi superiori, devono avere accesso a una ulteriore interfaccia di amministrazione che consenta inserimento, modifica, visualizzazione ed eliminazione della totalità dei dati gestiti dal sistema, compresi i dati degli utenti stessi.

Ogni utente dell'applicazione, sia esso un utente registrato o un amministratore, deve afferire ad un gruppo di ricerca in quanto si vuole che ad ogni esperimento inserito sia possibile associare uno dei seguenti ambiti di visibilità:

- privata, quando si vuole che i dati siano visibili al solo utente proprietario;
- di gruppo, quando si vuole che i dati siano visibili a tutti gli utenti di un dato gruppo;
- pubblica, quando si vuole che i dati siano visibili a tutti gli utenti dell'applicazione.

Laddove con il termine visibilità si è intesa la possibilità, da parte degli utenti, di visualizzare e scaricare i dati di un esperimento. L'unica eccezione al sistema degli ambiti di visibilità deve essere costituita dagli amministratori, che in ogni caso devono poter visualizzare, modificare ed eliminare i dati.

Ogni qual volta un utente inserisce un nuovo esperimento esso ne assume la proprietà e quindi può:

- modificarne le proprietà, incluso il livello di visibilità;
- aggiungere ulteriori analisi o campioni o modificare quelli esistenti;

- decidere di indicizzare i dati normalizzati e le liste di geni regolati.

Si vuole, infine, che l'applicazione offra agli utenti la possibilità di effettuare ricerche tra i dati archiviati, al fine di conoscere come si comportano particolari geni, o classi di geni, in diversi esperimenti o tra i campioni di uno stesso esperimento.

Per quanto riguarda la descrizione di un esperimento, in accordo a MIAME, si deve tener conto di quanto segue:

- un esperimento realizzato con la tecnica del microarray può essere realizzato su diversi array;
- ogni esperimento consta di una parte “teorica” o “centrale”, comune a tutto l’esperimento, ed una parte riguardante i campioni analizzati e le analisi statistiche sui dati;
- in un esperimento vengono analizzati uno o più campioni e su di essi possono essere fatte una o più analisi;
- ogni analisi consta nel confronto tra due campioni di uno stesso esperimento fissato un test statistico e il valore assunto da un dato parametro statistico;
- analisi e campioni nell'ambito dello stesso esperimento possono differire tra loro per i protocolli e le metodologie applicate.

Si da per scontato che la descrizione dell'esperimento debba essere corredata di tutti i meta dati necessari ai fini della comprensione e della ripetibilità dello stesso. Allo stesso modo, anche campioni e analisi devono essere accompagnati da un insieme definito di meta dati. Di tali meta dati si darà notizia nel modello concettuale.

2.2 Costruzione del Modello Concettuale dei Dati

Dovendo progettare una applicazione data-centrica, ci è sembrato naturale partire dalla costruzione di un Modello Concettuale dei Dati. Esso consiste in una rappresentazione di concetti (entità) e relazioni coinvolti nel dominio dell'applicazione.

2.2.1 Notazione

Nel nostro caso, avendo costruito il modello utilizzando Power Designer 12, la notazione utilizzata è del tipo “crow's feet”, così detta perché utilizza simboli che ricordano zampe di uccello. Tale notazione, comune ad altri tool di progettazione come Microsoft Visio, rappresenta le relazioni con linee di connessione tra le entità e utilizza simboli agli estremi di dette linee per rappresentare le cardinalità delle relazioni stesse.

Per rappresentare le cardinalità vengono utilizzati tre simboli:

- il cerchietto rappresenta "zero";
- il trattino rappresenta "uno";
- la zampa di corvo rappresenta "molti".

Questi simboli sono utilizzati in combinazione per rappresentare i quattro tipi di cardinalità che una entità può avere in una relazione:

- cerchietto → zero o uno;
- trattino → esattamente uno;
- cerchietto e zampa di corvo → zero o molti;
- trattino e zampa di corvo → uno o molti.

L'utilizzo di questa notazione comporta i seguenti benefici:

- chiarezza nell'identificare il lato molti della relazione, grazie alla zampa di corvo;
- una notazione sintetica per identificare le relazioni obbligatorie, utilizzando il trattino perpendicolare, o le relazioni opzionali, utilizzando il cerchietto.

2.2.2 Note sulla ristrutturazione del modello

Power designer 12, il tool da noi utilizzato, sollecita il progettista a realizzare un modello concettuale già ristrutturato in quanto offre la possibilità di tradurre automaticamente il modello concettuale in modello logico. Per utilizzare tale funzionalità, che utilizza le regole di traduzione del modello concettuale in modello logico, è opportuno anticipare la fase di ristrutturazione.

In sostanza, il modello da noi proposto a seguire non è un modello concettuale nel senso stretto del termine ma è un modello concettuale prodotto allo scopo di semplificare la traduzione e ottimizzare le prestazioni.

Riportiamo di seguito alcune delle operazioni effettuate:

- sono stati eliminati gli attributi multivalore trasformandoli in nuove entità in relazione con l'entità cui appartenevano;
- le entità utente registrato e amministratore sono state accorpate nell'unica entità “user”, dotando questa entità dell'attributo “accounttype” che consente la distinzione tra i due casi;
- alcuni attributi, di cui si voleva restringere il range di valori per uniformare il contenuto del data base e rendere più semplice la gestione dei dati, sono stati trasformati in entità atomiche dotate del solo attributo “name”;
- tutte le entità del modello sono state dotate di un identificativo primario “id” di tipo serial, cioè un intero il cui valore viene incrementato automaticamente alla creazione di ogni occorrenza.

In fine è da notare come ad ogni attributo sia già stato associato un dominio, cioè l'insieme di valori che esso può assumere, utilizzando i “data type” messi a disposizione dal tool di progettazione.

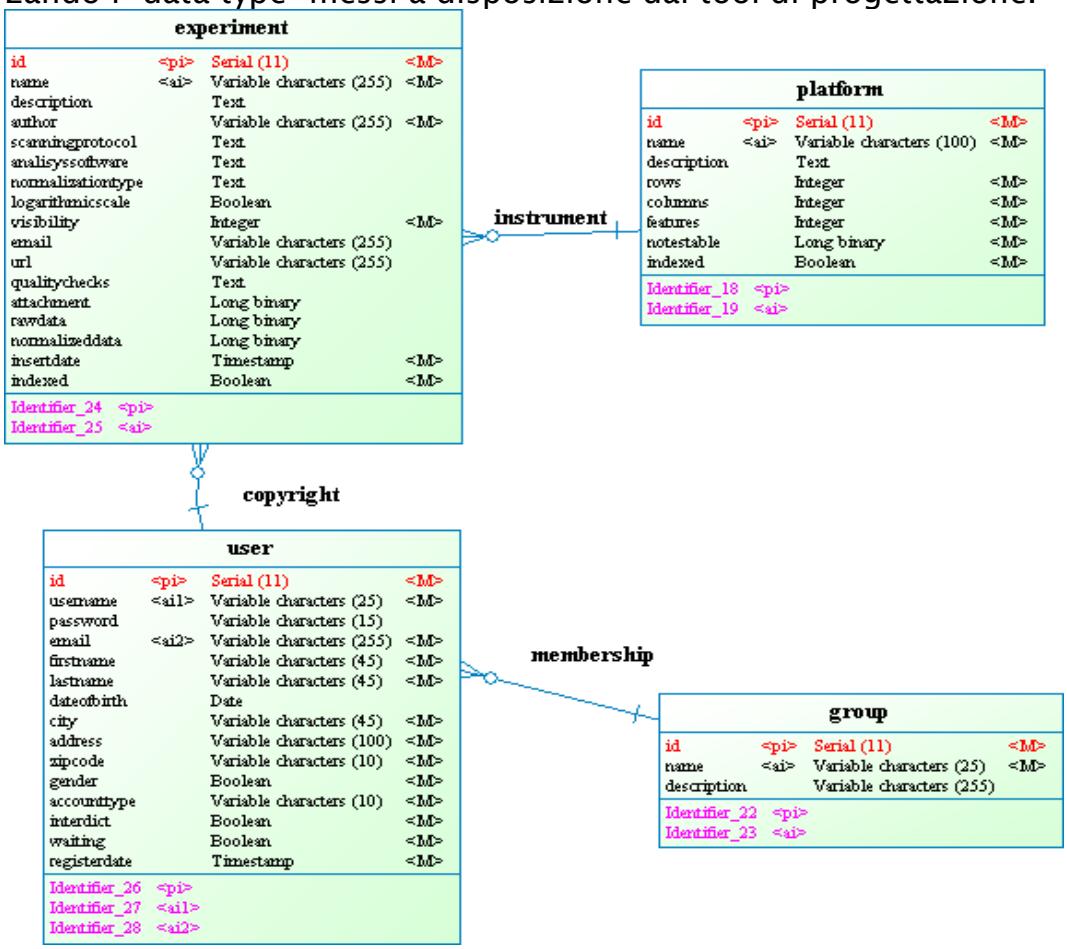

Figura 2.1.: Modello Concettuale dei Dati, vista semplificata.

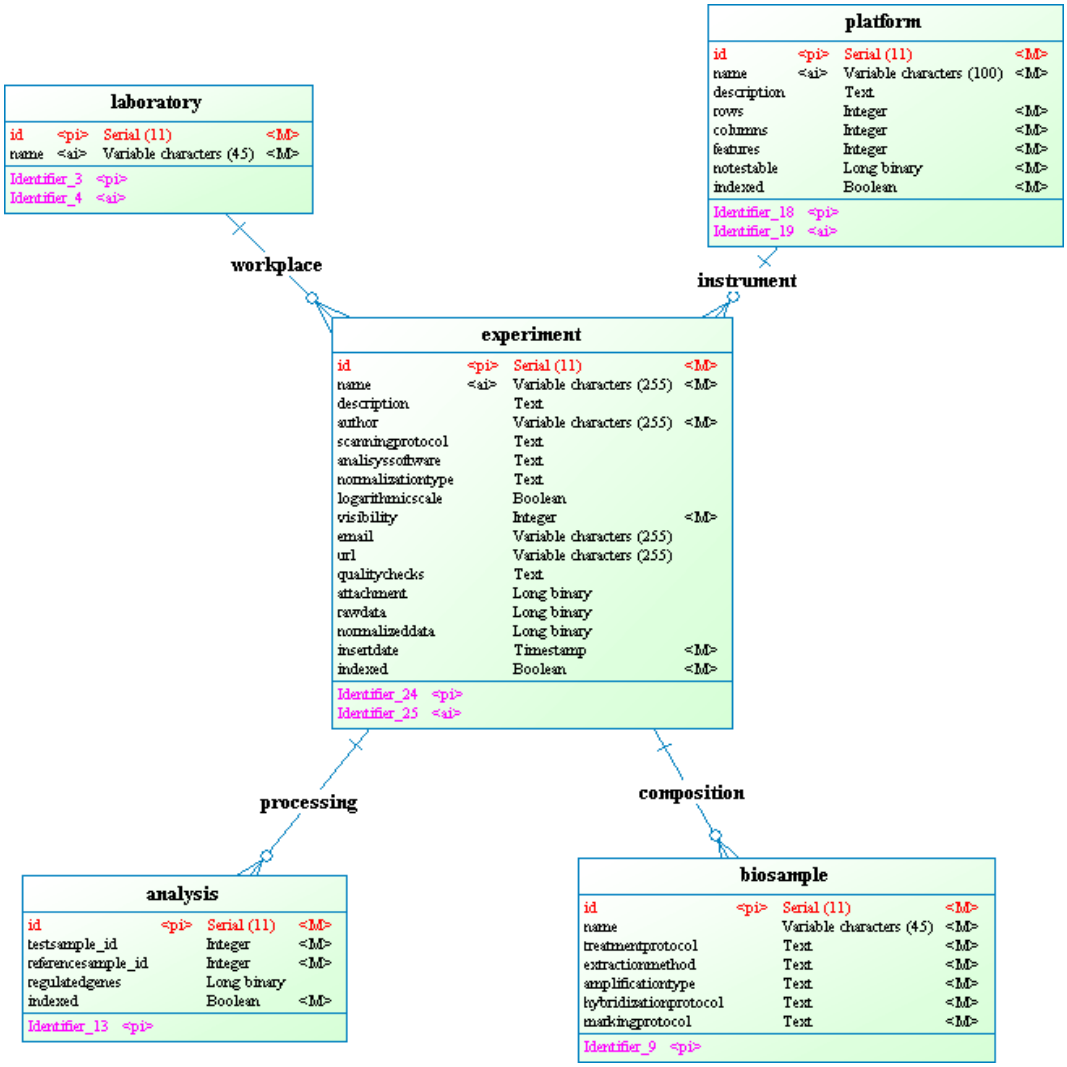

Figura 2.2.: Modello Concettuale dei Dati, modello semplificato di esperimento.

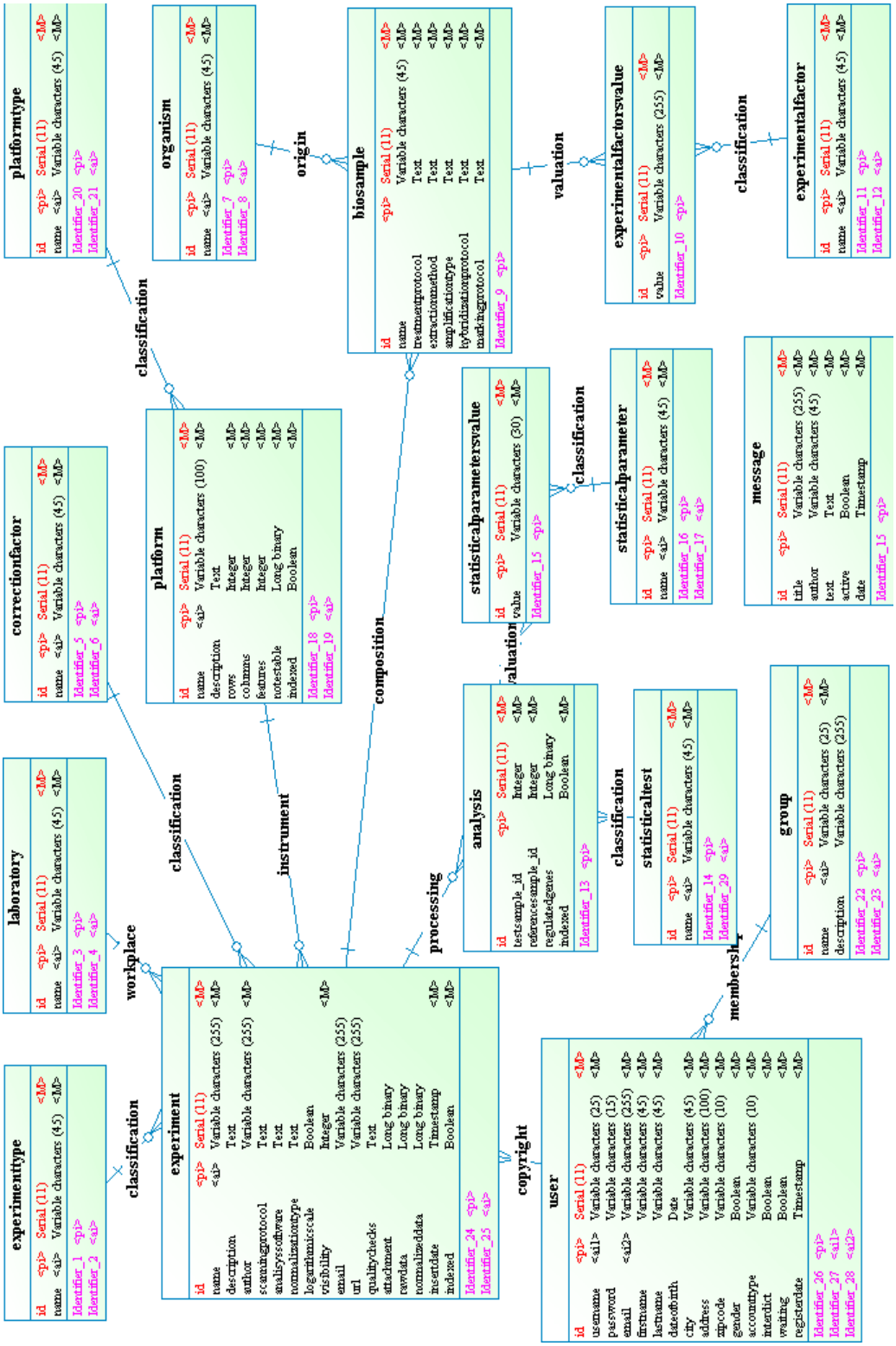

Figura 2.3: Modello Concettuale dei Dati, vista d'insieme.

2.3 Ulteriori vincoli sui dati

Il dominio applicativo pone ulteriori vincoli non esprimibili con la sintassi del modello entità relazione:

- i campioni legati ad uno stesso esperimento devono avere valori diversi dell'attributo “nome”;
- la relazione uno a molti tra gli attributi “testsample_id” e “referencesample_id” dell'entità analisi e l'attributo “id” dell'entità “biosample” non è stata codificata nel modello;
- poiché ogni analisi consta nel confronto tra due campioni di uno stesso esperimento, i valori assunti da “testsample_id” e “referencesample_id” devono essere scelti tra gli “id” dei “biosample” legati ad uno stesso “experiment”;
- non è possibile associare analisi ad un esperimento che non sia composto da almeno un campione.

2.4 Dizionario dei dati

2.4.1 Entità

Entità	Descrizione	Attributi
analysis	Una analisi consta nel confronto tra due campioni di uno stesso esperimento, fissato il valore di un determinato insieme di parametri statistici. Ad ogni analisi corrisponde una tabella contenente i geni regolati sotto quelle particolari condizioni.	id testsample_id referencesample_id regulatedgenes indexed
biosample	E' il materiale biologico dal quale sono stati estratti gli acidi nucleici per una successiva marcatura e ibridazione, scelti i valori di un determinato insieme di fattori sperimentali. MIAME distingue tra origine del campione (biosource), il suo trattamento, la preparazione dell'estratto e la sua marcatura.	id name treatmentprotocol extractionmethod amplificationtype hybridizationprotocol markingprotocol
correctionfactor	Specifica il fattore di correzione utilizzato nell'effettuare un esperimento.	id name
experiment	Questa entità rappresenta ogni esperimento di cui si vogliono archi-	id name description

	viare i dati. Ad ogni esperimento va associato un file zip contenente tutti i dati grezzi e una tabella (in formato di file di testo tabulato) contenente i dati normalizzati. Si da all'utente la possibilità di archiviare con l'esperimento anche un allegato generico, sempre compresso in un file zip, contenente ulteriori meta dati.	author scanningprotocol analysissoftware normalizationtype logarithmicscale visibility email url qualitychecks attachment rawdata normalizeddata insertdate indexed
experimentalfactor	Rappresenta il fattore sperimentale utilizzato nell'effettuare una analisi.	id name
experimentalfactorvalue	Rappresenta il valore del fattore sperimentale utilizzato nell'effettuare una analisi.	id value
experimenttype	Specifica la tipologia dell'esperimento.	id name
group	Rappresenta un gruppo di ricerca, cioè un sottoinsieme degli utenti del sistema.	id name description
laboratory	Rappresenta il laboratorio in cui vengono portati a termine gli esperimenti.	id name
message	Rappresenta un semplice messaggio di testo da mostrare nella home page per costruire un rudimentale sistema di gestione delle news.	id title author text active date

organism	Il gene e le specie e sottospecie dell'organismo dal quale è derivato il biosample (campione di biomateriale).	id name
platform	Questa entità rappresenta lo specifico array (il chip) sul quale è stato eseguito l'esperimento.	Id name description rows columns features notestable indexed
platformtype	Rappresenta la marca o tipologia di array utilizzato.	id name
statisticalparameter	Rappresenta il parametro statistico utilizzato nell'effettuare una analisi.	id name
statisticalparametervalue	Rappresenta il valore di un parametro statistico utilizzato in una analisi.	Id value
statisticaltest	Rappresenta il test statistico scelto nell'effettuare una analisi.	id name
user	Rappresenta l'utilizzatore del sistema, sia esso un utente registrato o un amministratore. Viene identificato tramite l'immissione della coppia username e password.	id username password email firstname lastname dateofbirth city address zipcode gender accounttype interdict

		waiting registerdate

2.4.2 Relazioni

Relazione	Descrizione	Componenti	Cardinalità
composition	Un esperimento è composto da zero o più campioni	experiment biosample	uno a molti
copyright	Un esperimento appartiene ad un unico utente	user experiment	uno a molti
instrument	L'esperimento è realizzato utilizzando un unica piattaforma	platform experiment	uno a molti
membership	Un utente appartiene ad un unico gruppo	goup user	uno a molti
origin	Un campione proviene da un unico organismo	organism biosample	uno a molti
processing	Sui dati di un esperimento possono essere effettuate zero o più analisi	experiment analisis	uno a molti
workplace	Un esperimento è realizzato in un unico laboratorio	laboratory experiment	uno a molti

Alle relazioni descritte in tabella vanno aggiunte le relazioni uno a molti che collegano le entità centrali del modello a tutte quelle entità dotate della sola coppia di attributi “id”, “name”. A tali relazioni è stato dato il nome generico “classification” e la loro interpretazione può essere dedotta semplicemente dal modello.

2.4.3 Attributi

Nelle tabelle contenute nei paragrafi a seguire sono specificati maggiori dettagli sugli attributi di ciascuna entità. Per ogni attributo è riportato:

- il nome;
- il Data Type, cioè il tipo di dato, scelto in fase di progettazione concettuale tra quelli disponibili in PowerDesigner;
- una breve descrizione testuale che indichi la proprietà del mondo reale che esso vuole modellare;
- un segno di spunta che indichi se l'attributo è obbligatorio (M sta per Mandatory), se è un identificatore (I sta per Identifier) o se è un identificatore primario (PI sta per Primary Identifier).

2.4.3.1 Attributi dell'entità analysis

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√	√	√
testsample_id	Integer	id del biosample di prova considerato nell'incrocio dei campioni per la descrizione delle analisi statistiche effettuate sull'esperimento	√		
referencesample_id	Integer	id del biosample di riferimento utilizzato nell'analisi	√		
regulatedgenes	Long binary	Lista dei geni selezionati ottenuti dal confronto dei due biosample in base a test e analisi statistiche			
indexed	Boolean	Booleano che assume valore vero se la lista	√		

Nome	Data Type	Descrizione	M	I	PI
		dei geni regolati è stata indicizzata da Lucene, falso altrimenti.			

2.4.3.2 Attributi dell'entità biosample

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√	√	√
name	Variable characters (45)	Nome del biosample. Biosamples relativi allo stesso esperimento devono avere nomi distinti.	√		
treatmentprotocol	Text	Il tipo di manipolazione applicata al biomateriale con gli obiettivi di generare una delle variabili sotto studio	√		
extractionmethod	Text	Il protocollo usato per estrarre gli acidi nucleici dal biosample	√		
amplificationtype	Text	Metodo usato per amplificare l'acido nucleico estratto	√		
hybridizationprotocol	Text	Documentazione della serie di passaggi eseguiti nell'ibridazione, inclusi: soluzione, agente bloccante e concentrazione	√		

		usati; tempo; concentrazione; volume; temperature e descrizione degli strumenti di ibridazione			
markingprotocol	Text	Protocollo di marcatura utilizzato	√		

2.4.3.3 Attributi dell'entità correctionfactor

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√	√	√
name	Variable characters (45)	Nome del fattore di correzione	√	√	

2.4.3.4 Attributi dell'entità experiment

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√	√	√
name	Variable characters (255)	Un nome associato all'esperimento che ne descriva brevemente la natura	√	√	
description	Text	Testo libero di descrizione dettagliata dell'esperimento con possibilità di aggiungere annotazioni, risorse, link a pubblicazioni etc.			
author	Variable characters (255)	Autore dell'esperimento. L'autore può differire dall'utente che inserisce l'esperimento e che, quindi, ne assume la	√		

		proprietà			
scanningprotocol	Text	Protocollo di scansione utilizzato, cioè la documentazione del set di passaggi eseguiti per scannerizzare l'array e generare un immagine, includendo una descrizione degli strumenti di scannerizzazione e dei settaggi dei parametri			
analisyssoftware	Text	Documentazione del set di passaggi eseguiti per quantificare l'immagine inclusi, il software di analisi, l'algoritmo e tutti i parametri usati			
normalizationtype	Text	Tipo di normalizzazione applicato			
logarithmicscale	Boolean	Booleano che assume valore vero, se è stata utilizzata una scala logaritmica, falso altrimenti.			
visibility	Tiny int	Visibilità dell'esperimento. Può assumere tre diversi valori associati a: • privato, • di gruppo, • pubblico	√		
email	Variable characters (255)	Email cui ci si può rivolgere per ulteriori informazioni sull'esperimento			
url	Variable characters (255)	Url dell'organizzazione, laboratorio, gruppo di ricerca o pagina personale di chi è informato sull'esperimento			

qualitychecks	Text	Misure eseguite per garantire o misurare la qualità: replicati(numero e descrizione) o altre			
attachment	Long binary	Allegato generico. Un file compresso contenente all'interno documentazione aggiuntiva a quella archiviabile nel sistema			
rawdata	Long binary	I dati prodotti dalla scansione delle immagini dell'array. Si suppone che l'utente abbia raccolto tali dati in un unico file compresso e che li sottometta al sistema in tale formato.			
normalizeddata	Long binary	I dati ottenuti dalle procedure di normalizzazione. Si suppone che essi siano raccolti in un file di testo tabulato.			
insertdate	Timestamp	La data di inserimento dell'esperimento, aggiornata ogni qual volta siano applicate modifiche	√		
indexed	Boolean	Booleano che assume valore vero se la lista dei geni regolati è stata indicizzata da Lucene, falso altrimenti.	√		

2.4.3.5 Attributi dell'entità experimentalfactor

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√

name	Variable characters (45)	Nome descrittivo di parametri o condizioni testate nell' l'esperimento	√	√	

2.4.3.6 Attributi dell'entità experimentalfactorsvalue

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
value	Variable characters (255)	Valore assunto da un fattore sperimentale	√		

2.4.3.7 Attributi dell'entità experimenttype

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
name	Variable characters (45)	Tipo di esperimento, per esempio: • normal vs. diseased comparison • treated vs. untreated comparison • time course • dose response • effect of gene knock-out • effect of gene knock-in (transgenics) • ...	√	√	

2.4.3.8 Attributi dell'entità group

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
name	Variable characters (25)	Nome del gruppo di ricerca o del gruppo di utenti	√	√	
description	Variable	Descrizione estesa e dettagliata			

	characters (255)	del gruppo di ricerca			

2.4.3.9 Attributi dell'entità laboratory

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
name	Variable characters (45)	Nome identificativo del laboratorio in cui si effettua la sperimentazione	√	√	

2.4.3.10 Attributi dell'entità message

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
title	Variable characters (255)	Titolo visualizzato del messaggio	√		
author	Variable characters (45)	Autore del messaggio. Può essere diverso dall'amministratore che provvede ad inserirlo	√		
text	Text	Testo contenuto nel messaggio	√		
active	Boolean	Variabile booleana che, se vera, indica che il messaggio deve essere mostrato nella home page, se falsa, altrimenti	√		
date	Timestamp	Data di inserimento del messaggio	√		

2.4.3.11 Attributi dell'entità organism

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
name	Variable	Testo descrittivo del gene e delle	√	√	

	characters (45)	specie e sottospecie dell'organismo dal quale è derivato il biomateriale			

2.4.3.12 Attributi dell'entità platform

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
name	Variable characters (100)	Nome dello specifico array sul quale è stato eseguito l'esperimento	√	√	
description	Text	Descrizione testuale dell'array			
rows	Integer	Numero di righe dell'array	√		
columns	Integer	Numero di colonne dell'array	√		
features	Integer	Numero di reporters sull'array	√		
notestable	Long binary	Tabella fornita dal produttore dell'array. Si suppone che l'utente la immetta nel sistema sotto forma di file di testo tabulato	√		
indexed	Boolean	Variabile booleana che, se vera, indica che la tabella delle annotazioni è stata indicizzata da Lucene, se falsa altrimenti	√		

2.4.3.13 Attributi dell'entità platformtype

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
name	Variable characters (45)	Nome o marca della specifica piattaforma su cui viene effettuata la sperimentazione	√	√	

2.4.3.14 Attributi dell'entità statisticalparameter

Nome	Data Type	Descrizione	M	I	PI

id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
name	Variable characters (45)	Nome del parametro statistico	√	√	

2.4.3.15 Attributi dell'entità statisticalparametersvalue

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
value	Variable characters (30)	Valore assunto dal particolare parametro statistico	√		

2.4.3.16 Attributi dell'entità statisticaltest

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
name	Variable characters (45)	Nome del test statistico	√	√	

2.4.3.17 Attributi dell'entità user

Nome	Data Type	Descrizione	M	I	PI
id	Serial (11)	Identificativo primario di tipo intero ad incremento automatico	√		√
username	Variable characters (25)	Nome con cui l'utente viene riconosciuto dal sistema. Per ragioni di sicurezza deve avere una lunghezza minima.	√	√	
password	Variable characters	Password scelta dall'utente per il riconoscimento. Per ra-			

	(15)	gioni di sicurezza deve avere una lunghezza minima			
email	Variable characters (255)	Email dell'utente	√	√	
firstname	Variable characters (45)	Nome reale dell'utente	√		
lastname	Variable characters (45)	Cognome dell'utente	√		
dateofbirth	Date	Data di nascita			
city	Variable characters (45)	Città di residenza o luogo di lavoro	√		
address	Variable characters (100)	Indirizzo di residenza o di lavoro	√		
zipcode	Variable characters (10)	CAP del luogo di residenza o di lavoro	√		
gender	Boolean	Variabile booleana che identifica il sesso dell'utente: maschio o femmina	√		
accounttype	Boolean	Tipo di account dell'utente registrato: utente o amministratore.	√		
interdict	Boolean	Variabile booleana che specifica se l'utente è stato interdetto dall'utilizzo del sistema	√		
waiting	Boolean	Variabile booleana che specifica se l'utente è in attesa della validazione di uno degli amministratori. Il valore assunto di default è “true”	√		
registerdate	Timestamp	Data di registrazione dell'utente	√		

2.5 Costruzione del modello fisico dei dati

A partire dal modello concettuale prodotto, scelto il dbms da utilizzare, PowerDesigner 12 applica le opportune regole di traduzione per proporre all'utente un modello fisico (o anche logico) dei dati.

Così, scelto MySQL 5.0, PowerDesigner ha prodotto il modello fisico in figura 2.4. Nel modello fisico dei dati è evidente come PowerDesigner abbia applicato le seguenti regole di trasformazione:

- le entità sono state trasformate in tabelle includenti gli stessi attributi;
- gli identificatori primari delle entità sono diventati le chiavi primarie delle tabelle;
- gli ulteriori identificatori, dove presenti, sono diventati indici di tipo “unique”;
- le relazioni uno a molti sono state tradotte con l'assorbimento nell'entità lato uno della relazione e l'aggiunta all'entità lato molti della chiave primaria del lato uno;
- le relazioni molti a molti sono state tradotte in tabelle contenenti tra gli attributi le chiavi primarie delle due entità in gioco;

Nel modello sono stati inoltre introdotti vincoli di integrità referenziale in corrispondenza delle chiavi straniere (foreign key, in azzurro nel modello). Tali vincoli sono stati impostati, per comodità e sicurezza, nel modo seguente:

- ON DELETE NO ACTION,
- ON UPDATE CASCADE.

In questa maniera non è possibile cancellare, nemmeno accidentalmente, righe la cui chiave primaria sia utilizzata come chiave esterna in un'altra tabella mentre, in caso di update della chiave pri-

maria, la modifica viene ereditata dalle righe di tutte le tabelle in cui questa è chiave esterna.

Per poter utilizzare i vincoli di integrità referenziale in MySQL è stato necessario scegliere il table engine InnoDB per tutte le tabelle generate, in luogo del predefinito MyISAM, che non supporta tale funzionalità.

In fine, sempre tramite PowerDesigner, si è provveduto alla generazione dello script di creazione del database. Tale script altro non è che un file di testo contenente un insieme di comandi in linguaggio SQL per la creazione delle tabelle, delle viste e dei vincoli.

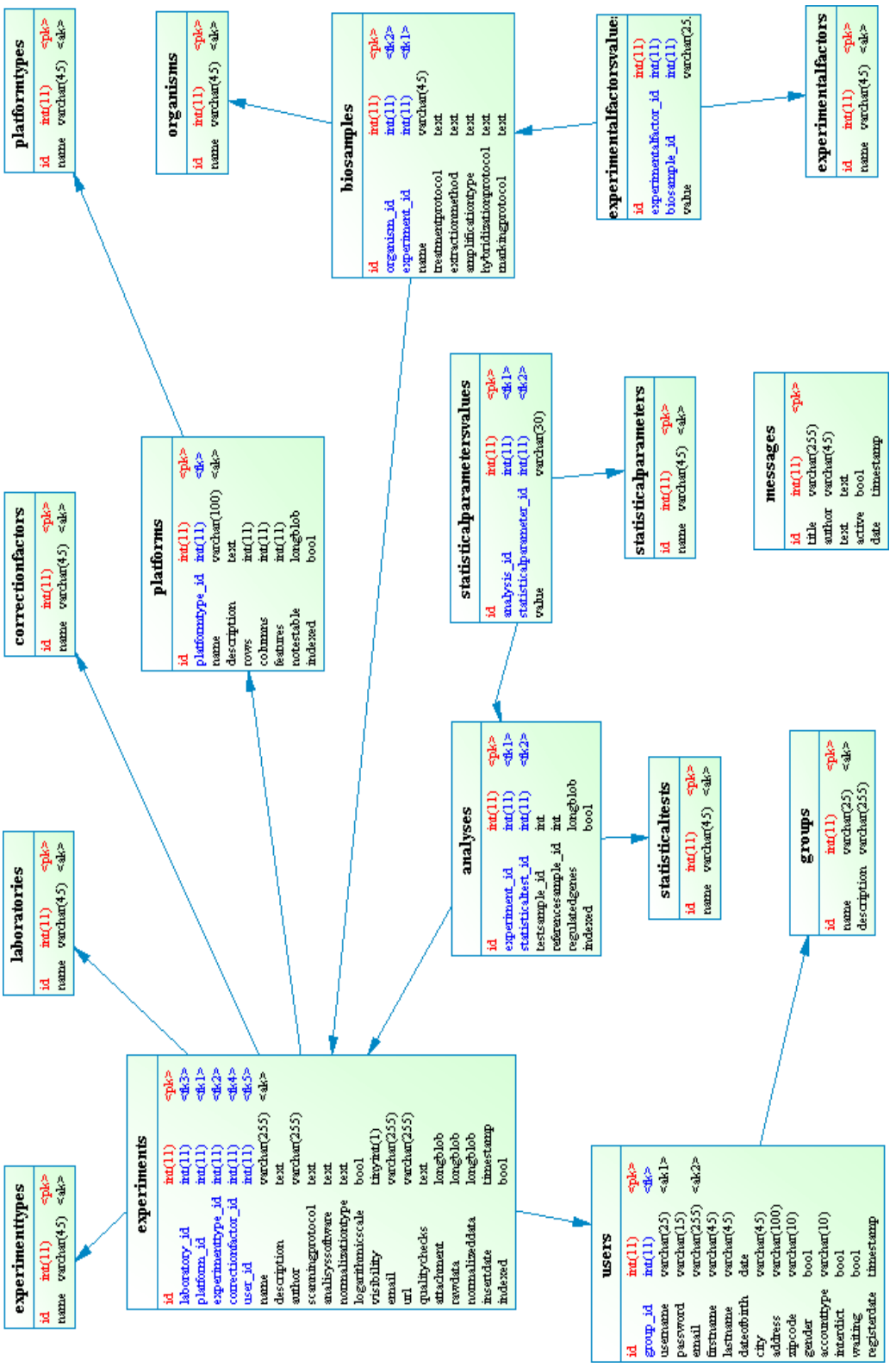

Figura 2.4.: Modello Fisico dei Dati.

3 Progettazione dell'applicazione

Contenuto del capitolo

3.1 Premessa

Parallelamente alla progettazione dei dati, e quindi alla definizione dell'aspetto con cui si sarebbe presentato il database, è stata portata avanti la progettazione dell'applicazione vera e propria. Durante il processo di sviluppo spesse volte il modello dei dati è stato rimaneggiato per far fronte a nuovi requisiti posti dal committente o per superare difficoltà incontrate nella fase di sviluppo. I due modelli, quello dei dati e quello dell'applicazione, hanno seguito cicli di sviluppo distinti, paralleli ma non per questo dissociati. E' ovvio come i requisiti definiti nella fase di modellazione dei dati siano un sottoinsieme dei requisiti cui il software deve rispondere. Mentre nella fase di progettazione dei dati si è posto l'accento sul problema della persistenza, in questa fase successiva le nostre attenzioni si sono focalizzate attorno alle funzionalità che l'applicazione doveva offrire ai suoi utenti.

3.2 Iterative and incremental development

Tale attività è stata svolta seguendo un modello di sviluppo del software incrementale ed iterativo, una alternativa al classico modello a cascata (Waterfall Model) che cerca di superarne le più evidenti debolezze. Il modello scelto ha il vantaggio di adattarsi bene ad un lavoro di tesi, un lavoro nel corso del quale è implicito l'allargamento delle conoscenze e, in definitiva, l'allargamento delle nostre capacità di sviluppo.

Il modello di sviluppo iterativo ed incrementale è un modello di sviluppo ciclico che ruota attorno ai due seguenti concetti:

- Lo sviluppo incrementale è una strategia di programmazione e allestimento in cui le varie parti del software sono sviluppate in momenti differenti, e integrate una volta complete. Per esempio una release incrementale può includere nuove funzionalità, una nuova interfaccia, nuovi algoritmi, nuove funzionalità o nuove tecnologie. Ciò non implica e non preclude lo sviluppo iterativo o a cascata perché queste sono utilizzate come strategie di rimodellazione;
- Lo sviluppo iterativo è utilizzato come strategia di programmazione della rimodellazione in cui il tempo è speso per rivisitare e migliorare parti del sistema. Ciò non presuppone uno sviluppo incrementale, ma lavora bene con esso.

Una tipica differenza tra i due approcci è che l'output di un incremento è rilasciato agli utenti, mentre l'output di una iterazione è esaminato per eventuali modifiche. In realtà nel linguaggio del software engineering spesso si parla di sviluppo iterativo intendendo una qualsiasi combinazione di sviluppo iterativo e incrementale, così che spesso si utilizza uno solo dei due termini per indicare un processo di sviluppo che ha fatto uso di entrambe le pratiche.

Seguendo tale modello, nel corso del processo di sviluppo di questa nostra applicazione web, più versioni sono state rilasciate agli utenti su medeaserver.isa.cnr.it e, mentre si procedeva allo sviluppo di nuove funzionalità, si riceveva l'output degli utenti riguardo a quelle già sviluppate.

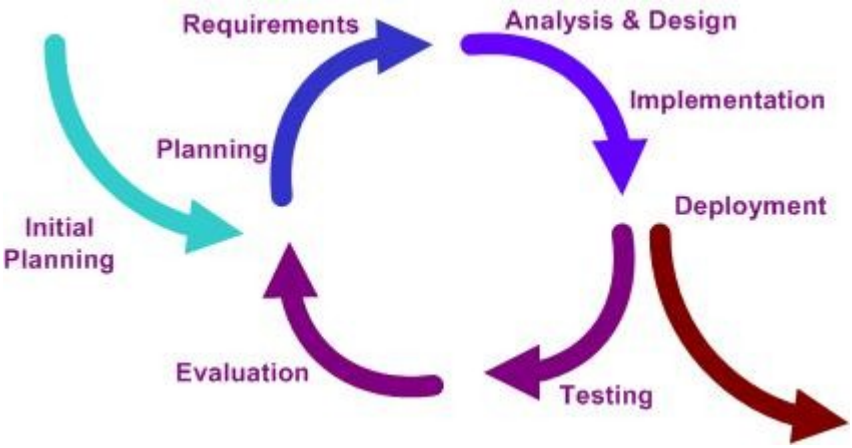

Figura 3.1.: Iterative and Incremental Development's lifecycle.

L'idea di base dietro questo processo di miglioramento iterativo è quella di sviluppare il sistema software in modo incrementale, consentendo allo sviluppatore di avvantaggiarsi di quanto appreso durante lo sviluppo delle versioni preliminari del sistema. Ove possibile l'apprendimento dovrebbe provenire sia dallo sviluppo che dall'utilizzo del sistema. I passi chiavi del processo prevedono che si parta da una semplice implementazione di un sottoinsieme dei requisiti per poi migliorare iterativamente le versioni prodotte fino a che il sistema non è stato pienamente implementato. Ad ogni iterazione vengono apportate modifiche al progetto e aggiunte nuove funzionalità.

La procedura in se ruota attorno a:

- step di inizializzazione,
- step di iterazione,
- project control list.

Il passo di inizializzazione crea una versione base del sistema con l'obbiettivo di creare un prodotto con cui l'utente possa interagire. Esso dovrebbe offrire una campionatura degli aspetti chiave del problema e fornire una soluzione che sia abbastanza semplice da comprendere e implementare. Una project control list, semplice lista di tutti i task che devono essere svolti, viene redatta per guidare il processo di iterazione. Tale lista contiene caratteristiche da implementare, e le modifiche da apportare al modello precedentemente creato. La project control list viene costantemente rivisitata al termine di ogni fase di analisi dei requisiti.

Ogni iterazione consiste nella riprogettazione ed implementazione di un task proveniente dalla project control list e nell'analisi della versione corrente del sistema. L'obbiettivo che ci si pone è quello di ottenere le nuove funzionalità in maniera semplice, diretta e soprattutto modulare. L'analisi in ogni iterazione è basata sul feedback fornito dall'utente e sullo studio delle funzioni già disponibili. In fine, la project control list viene ricompilata alla luce del risultato di ogni fase di analisi.

3.3 Breve cronistoria del processo di sviluppo

Il punto di partenza del processo di sviluppo, come già detto, è stato il lavoro svolto tempo addietro presso i laboratori del CNR di Avellino. Ci è stato posto il problema di riprogettare completamente l'applicazione sostituendo al framework utilizzato da chi ci ha preceduti, Java Server Faces, il più anziano e diffuso Apache Struts. In sostanza si voleva un applicativo completamente nuovo che replicasse le principali funzionalità dell'applicazione web precedentemente sviluppata e le estendesse introducendo funzionalità di ricerca full text tra i dati degli esperimenti.

Il lavoro preesistente di raccolta e analisi dei requisiti è stato ereditato dal nostro progetto in massima parte. In particolare sono risultati molto utili il glossario dei termini e l'SRS (Software Requirements Specification) allegati a tale lavoro, perchè hanno velocizzato la fase di raccolta dei requisiti.

In una successiva fase del processo di sviluppo è stata studiata la possibilità di riutilizzare parte della business logic dell'applicativo preesistente ma, da un lato l'utilizzo di un framework differente, dall'altro aggiustamenti al modello, ci hanno spinto a riprogettare il sistema in toto. E' stato subito chiaro, infatti, come la scelta di utilizzare un framework influenzi largamente le fasi della modellazione e dello sviluppo. Ciò sarà evidente quando mostreremo i class diagram disegnati per il nuovo sistema.

I primi prototipi sviluppati hanno riguardato la gestione degli utenti e dei privilegi. Meccanismi per la registrazione, il login, il logout e l'attivazione degli account sono componenti ormai standard di una applicazione web e sono quindi stati progettati e realizzati indipendentemente dal resto dell'applicazione.

Mentre si testava la gestione degli utenti e dei privilegi è partita la realizzazione dell'interfaccia grafica di CRUD[1] dei dati gestiti dal DBMS. Tali funzionalità sono state sottoposte agli utenti committenti per avere un riscontro riguardo alla semplicità di utilizzo, alla semplicità di accesso e gestione dei dati. Siamo partiti dallo sviluppo del CRUD dei dati posti alla periferia del modello e ci siamo spostati verso il centro correggendo di volta in volta quanto necessario al raggiungimento dei requisiti specificati in partenza.

Una volta giunti alla realizzazione di un applicativo che replicasse le funzionalità del precedente, che consentisse cioè la gestione di un repository online di esperimenti, è stato affrontato il problema della ricerca. Ciò ha dato il via ad un ulteriore ciclo di sviluppo centrato attorno al problema dell'indicizzazione e del recupero dei dati. Come conseguenza del processo di sviluppo adottato, il componente che si occupa di ricerca e indicizzazione è disaccoppiato dal componente che si occupa dell'archiviazione dei dati col vantaggio di poter godere della sicurezza dell'uno e della scalabilità dell'altro.

Attualmente l'applicazione gira su medeaserver.isa.cnr.it ed è al vaglio degli utenti che ne stanno testando le funzionalità.

[1]CRUD sta per Create Retrieve Update & Delete e si riferisce alla classica interfaccia grafica per la manipolazione dei dati archiviati in un database.

3.4 Scelte tecnologiche

Un sottoinsieme degli aspetti tecnologici di questo lavoro sono stati determinati a monte del processo di progettazione, in conseguenza del confronto col relatore. E' stato fatto un rapido studio di fattibilità, fondato sulla ricerca di soluzioni tecnologiche in grado di favorire un rapido sviluppo e di fornire supporto ai requisiti attuali e a eventuali espansioni. Da esso è emersa la scelta delle seguenti tecnologie:

- il linguaggio di programmazione: Java;
- la piattaforma tecnologica: Java Enterprise Edition;
- l'application server: Apache Tomcat;
- il dbms: MySQL;
- il framework: Apache Struts.

Tra le principali motivazioni che hanno imposto tali scelte è bene sottolineare:

- la gratuità di tali prodotti e il fatto che fossero open source (ecczion fatta per la piattaforma Java);
- l'abbondante documentazione disponibile online;
- le rinomate affidabilità e robustezza della piattaforma Java e dei prodotti Apache;
- la preesistenza di Apache Tomcat e Mysql sul server a nostra disposizione;
- la disponibilità di un know-how consolidato riguardo a tali tecnologie nell'ambiente in cui è stato sviluppato il progetto.

Java è un linguaggio di programmazione prodotto da Sun Microsystems a partire dal 1995. Esso deriva la sua sintassi principalmente da C e C++ ma ha un più forte orientamento verso il mondo della programmazione ad oggetti. Una applicazione Java viene tipicamente compilata in un bytecode che può essere eseguito da una Java Virtual Machine, un middleware che si pone al di sopra del

sistema operativo rendendo di fatto Java indipendente dalla coppia hardware/sistema operativo. Il principale punto di forza di Java è la portabilità; il principale difetto le performance.

Java Enterprise Edition, o anche JavaEE, è una piattaforma per la programmazione lato server nel linguaggio di programmazione Java ampiamente utilizzata. La piattaforma JavaEE differisce dalla versione Standard Edition, o JavaSE, per la presenza di librerie addizionali che forniscono funzionalità per lo sviluppo di software fault-tolerant, distribuito e multi-tier basato sull'utilizzo di componenti modulari. Essa include un insieme di specifiche che definiscono le caratteristiche e le interfacce di tecnologie pensate per la realizzazione di applicazioni di tipo enterprise e mission critical. Chiunque può realizzare una implementazione di tali specifiche e produrre application server compatibili con le specifiche J2EE.

Apache Tomcat è un web container, o un application server, sviluppato da Apache Software Foundation, che implementa le specifiche Servlet e JavaServer Pages di Sun Microsystems. Esso fornisce, quindi, una piattaforma per l'esecuzione di applicazioni Web sviluppate nel linguaggio Java; include una interfaccia web per la gestione e la configurazione; mette a disposizione un proprio web server per la gestione di contenuti statici. L'utilizzo di Tomcat nella Reference Implementation ufficiale delle tecnologie JSP e Servlet, manifesta chiaramente le qualità del prodotto.

MySQL è un database management system relazionale multi-thread e multi-utente che, secondo MySQL AB, la società svedese che lo produce, vanta 10 milioni di installazioni. La versione base gira come un server che da accesso multi-utente alle basi di dati ospitate. Esso supporta un ampio sottoinsieme dei comandi ANSI SQL 99, oltre ad offrire proprie estensioni. Fornisce strumenti avanzati come i triggers, le stored procedures, le updatable view e il supporto transazionale su un ampio numero di piattaforme diverse.

MySQL è fornito di API per l'accesso ai dati scritte in molti linguaggi; offre una interfaccia ODBC, detta MyODBC e un driver JDBC. Il produttore distribuisce due utilissimi tool di amministrazione, MySQL Administrator e MySQL Query Browser che consentono l'accesso remoto, ma ve ne sono anche di sviluppati da terze parti come phpMyAdmin, una interfaccia web scritta in PHP. Una peculiarità di MySQL è che ogni tabella può essere gestita da uno storage engine diverso, a scelta tra quelli ufficiali e quelli distribuiti da terze parti. Gli storage engine possono differire per le prestazioni, per le funzionalità supportate, per le strategie di locking, per il supporto transazionale, ecc.

Apache Struts è un framework open source per lo sviluppo di applicazioni web su piattaforma J2EE. Struts estende le Java Servlet, incoraggiando gli sviluppatori all'utilizzo del pattern Model-View-Controller. L'utilizzo di Struts permette lo sviluppo di web application di notevoli dimensioni, agevolando la suddivisione dello sviluppo del progetto fra vari sotto-team. In altre parole, i progettisti, i grafici e i programmatori possono gestire in parallelo e autonomamente la loro parte del progetto. Tra le funzionalità offerte c'è l'internazionalizzazione, una potente tag library, un framework per la validazione dei form e un sistema per la gestione dei template.

3.5 Ambiente di sviluppo

Tutto il software oggetto del nostro lavoro è stato sviluppato utilizzando il noto Eclipse SDK[2], prodotto open source di Eclipse Foundation, e Exadel Studio Pro 4, un plug-in per Eclipse attualmente concesso in licenza gratuita da Exadel fino alla fine del 2007. Riportiamo a seguire alcune delle principali funzionalità offerte da Exadel Studio Pro:

- supporto allo sviluppo di software su JSF e Struts;
- editor visuale WYSIWYG[3] per JSF e Struts;
- scrittura assistita e riempimento automatico di codice JSP e XML;
- visualizzazione grafica del file di configurazione di Struts.

L'utilizzo di Exadel Studio Pro ci ha consentito di tenere in funzione, in un unico ambiente di sviluppo, sia l'applicazione preesistente, sviluppata su JSF, sia la nuova applicazione, sviluppata su Struts. In questo modo abbiamo potuto confrontare continuamente le funzionalità dell'applicazione preesistente e dell'applicazione nuova, sia per quanto riguarda l'interfaccia grafica (lato view), sia per quanto riguarda i dettagli implementativi (lato model).

Le funzionalità illustrate sopra si sono aggiunte alle funzionalità già esistenti in Eclipse SDK, tra cui:

- il completamento automatico del codice Java;
- la funzionalità “autobuild” con la sottolineatura degli errori;
- la vista integrata sulla console;
- il browser integrato;
- il servlet engine integrato;

[2]SDK sta per Software Development Kit, cioè strumento di sviluppo software.
[3]WYSIWYG sta per “what you see is what you get”, ciò che vedi è cio che ottieni.

- il debugger;
- la formattazione automatica del codice;
- la generazione automatica di metodi comuni quali getter, setter e costruttori.

3.6 Il framework utilizzato: Apache Struts 1.2.9

Prima di illustrare i diagrammi e i modelli che hanno guidato le fasi della progettazione è opportuno soffermarsi brevemente sul framework utilizzato. Un framework, infatti, non solo può guidare e supportare la fase implementativa, ma può anche orientare fortemente le fasi precedenti della progettazione. In particolare, avendo utilizzato Apache Struts 1.2.9, un framework MVC, è bene precisare brevemente anche in cosa consista il pattern MVC e come questo supporti la progettazione.

3.6.1 Il pattern MVC

MVC sta per Model-View-Controller e consiste in un modello architetturale (design pattern) utilizzato spesso nella progettazione del software. Nelle applicazioni più complesse, infatti, spesso gli svilupatori desiderano seraparare i dati, o meglio il modello dei dati (model), dall'interfaccia (view), in modo che i cambiamenti nell'interfaccia utente non intacchino la gestione dei dati, e che il modello dei dati possa essere riorganizzato senza modificare l'interfaccia stessa.

Il pattern MVC risolve questo problema disaccoppiando l'accesso ai dati e la business logic (model) dalla presentazione dei dati e interazione con l'utente (view) mediante l'introduzione di un nuovo componente intermedio, il controller.

3.6.1.1 Model

Il model consiste nella specifica rappresentazione delle informazioni su cui l'applicazione deve operare. Il modello MVC non indica con precisione come debba essere strutturata la business logic o come debba essere gestita la persistenza dei dati perchè è chiaro che tutto questo debba essere incapsulato nel modello.

3.6.1.2 View

La view raccoglie tutto quello che viene presentato all'utente. Essa esprime il modello in una forma adatta all'interazione con l'utente, tipicamente si tratta di una semplice interfaccia utente. Ad un singolo modello possono essere associate multiple view, per perseguire differenti scopi.

3.6.1.3 Controller

Il controllo processa e risponde agli eventi, tipicamente azioni dell'utente, e può invocare operazioni che modifichino il modello.

3.6.1.4 Tipica interazione dei tre componenti

Il modello MVC è spesso utilizzato nello sviluppo di applicazioni web, dove la view è una semplice pagina HTML e il controller è il componente che organizza e raccoglie dinamicamente i dati e genera il contenuto delle pagine stesse.In fine il modello è rappresentato dal contenuto delle pagine, cioè i dati, di solito organizzati in un database o in file XML.

Sebbene il modello MVC sia disponibile in diverse varianti e differenti implementazioni, in generale esso funziona così:

- L'utente interagisce con la view, l'interfaccia utente, in qualche modo (ad esempio premendo bottone, cliccando su un link o digitando un indirizzo);
- Il controller raccoglie l'evento proveniente dall'interfaccia e sceglie un gestore appropriato;
- Il gestore prescelto accede al model e richiede a quest'ultimo che sia attuato quanto richiesto dall'utente (ad esempio raccoglie i dati da un database, legge un file o invia un messaggio di posta elettronica);
- La view usa i dati ottenuti dal modello per generare una nuova interfaccia da fornire all'utente in risposta;
- L'interfaccia utente (view) attende ulteriori azioni dell'utente che diano inizio ad un nuovo ciclo.

In conclusione, disaccoppiando model e view, il pattern MVC aiuta a ridurre la complessità del disegno architetturale e a incrementare la flessibilità ed il riuso. Inoltre, questo non è il nostro caso, è bene notare come il pattern MVC, grazie al disaccoppiamento, aiuta nell'organizzazione di un lavoro di gruppo: persone differenti possono lavorare a componenti diversi, model e view, senza preoccuparsi del modo in cui interagiscono.

3.6.2 Il modello MVC in Apache Struts

Quando si dice che Struts è un framework MVC per lo sviluppo di applicazioni web, si vuole dire che esso facilita lo sviluppo rapido di applicazioni fornendo un controller che realizza l'interazione tra il model e la view. In questo modo il progettista non deve preoccuparsi di come rendere indipendenti il model e la view e tanto meno di come farli interagire. Fondamentale, quindi, è comprendere che Struts "è" il controller.

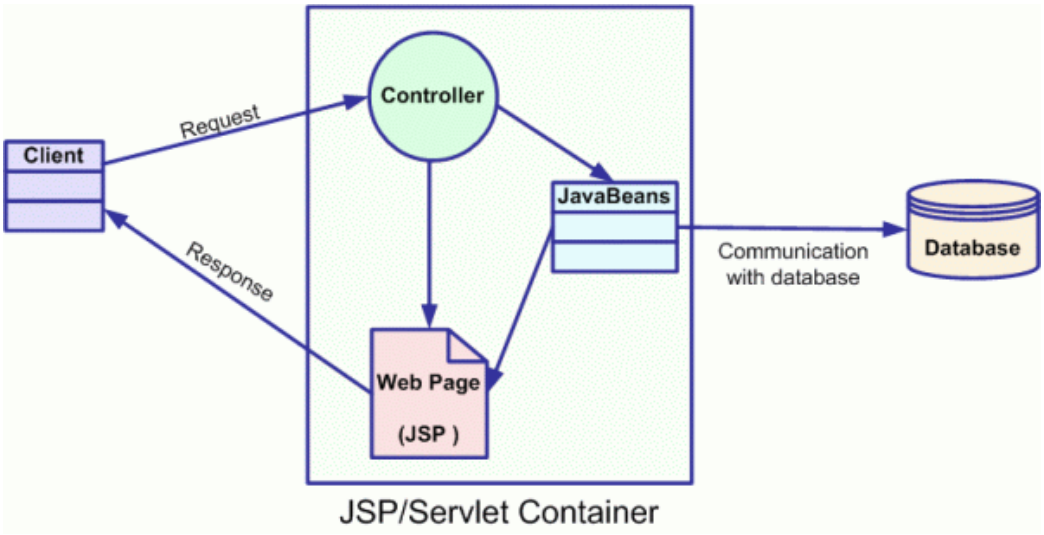

Figura 3.2.: Il modello MVC in Apache Struts.

Struts fornisce il suo controller e interagisce con altre tecnologie per fornire il model e la view. Per quanto riguarda il modello, Struts può interagire con le più diffuse tecnologie per l'accesso ai dati, come JDBC e EJB, o con packages prodotti da terze parti come Hibernate, iBATIS o Object Relational Bridge. Per quanto riguarda la view, invece, Struts lavora bene con le pagine JSP, con i Velocity Templates, XSTL, ed altri sistemi di presentazione. In figura 62 si può osservare una schematizzazione del funzionamento di Struts all'interno di un servlet container.

Nel nostro caso la scelta è caduta sulle soluzioni più semplici, diffuse e meglio documentate:

- Il model è stato sviluppato attraverso l'integrazione di JavaBeans e JDBC;
- La view è stata sviluppata sotto forma di pagine JSP utilizzando le tag libraries fornite da Struts e Display tag, una tag library open source.

3.6.3 I vantaggi nell'utilizzo di Struts

Oltre a fornire il proprio controller, Struts facilita la realizzazione di applicazioni web includendo un insieme di utilità che riducono i tempi di sviluppo e incrementano la maneggevolezza dell'applicazione. Tra esse ricordiamo:

- **La configurazione centralizzata in un file XML**
  Piuttosto che codificare informazioni di configurazione all'interno del programma, tali valori sono rappresentati in file XML o in file di proprietà. Questo disaccoppiamento implica che molti cambiamenti possono essere apportati senza modificare o ricompilare il codice Java, ma semplicemente modificando un singolo file. Questa caratteristica fa si che gli sviluppatori possano concentrarsi sui propri compiti specifici, senza il bisogno di conoscere tutto il layout del sistema;
- **Form-Beans**
  I form-beans facilitano lo scambio dei dati tra il model e la view. Essi raccolgono automaticamente i dati con cui l'utente popola i form e li rendono disponibili per l'elaborazione da parte del controller;
- **Tag libraries**
  Struts fornisce un insieme di librerie di tag JSP che facilitano la creazione della view. La libreria bean fornisce tag che facilitano l'output delle propietà dei JavaBeans; la libreria html, aiuta a creare form HTML da associare ai JavaBeans;
- **Validator**
  Validator è un framework per la validazione dei dati inseriti dagli utenti completamente integrato in Struts. Quando i va-

lori inseriti sono assenti o forniti in un formato improprio il form può essere ripresentato automaticamente all'utente con messaggi di errore. Tale validazione può essere effettuata lato server o lato client.

3.7 Use Case Diagrams

3.7.1 Actors

A valle del processo di raccolta e analisi dei requisiti è possibile dedurre quali saranno gli attori (actors) che dovranno interagire col nostro sistema. Nel formalismo UML un attore è qualcuno o qualcosa che fornisce uno stimolo al sistema. Un attore non può essere controllato dal sistema ed è definito come entità esterna al sistema stesso. Un attore può essere pensato come colui che svolge un ruolo, piuttosto che come una persona reale, così una singola persona del mondo reale può essere rappresentata da molti attori se essa interpreta ruoli differenti nell'interazione col sistema.

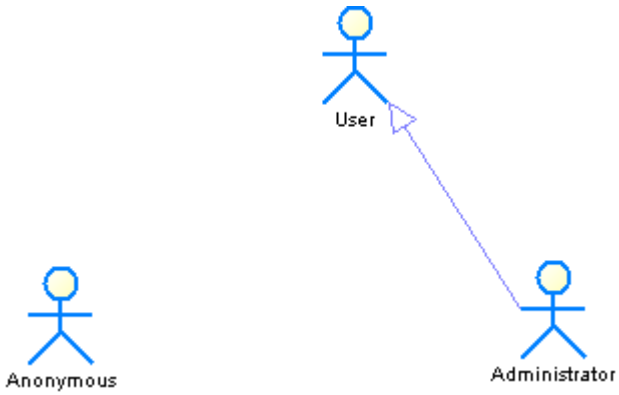

Figura 3.3.: Gli attori individuati.

Gli attori che abbiamo individuato sono sostanzialmente tre:

- **Anonymous**, il visitatore anonimo che accede alle pagine dell'applicazione senza autenticarsi;
- **User**, l'utente registrato, cioè l'utente che abbia completato il processo di identificazione;
- **Administrator**, l'amministratore del sistema.

Come si può notare dalla figura 3.3, l'amministratore altro non è che una specializzazione dell'attore utente. In sostanza le funzionalità a cui accede un utente sono un sottoinsieme delle funzionalità

cui ha accesso l'amministratore. Quest'ultimo deve avere il pieno controllo dei dati archiviati dall'applicazione, mentre il semplice utente registrato deve avere accesso solo ai dati disponibili nel suo ambito di visibilità.

Nelle descrizioni testuali dei casi d'uso utilizzeremo, abusando della pazienza del lettore, la seguente terminologia:

- **Anonymous** o **visitatore anonimo** o semplicemente **visitatore** per riferirci ai visitatori non registrati presso il sistema;
- **Utente** o **user** per riferirci al generico utente registrato presso il sistema;
- **Administrator** o **amministratore** per riferirci all'amministratore del sistema.

3.7.2 Use cases dell'attore Anonymous

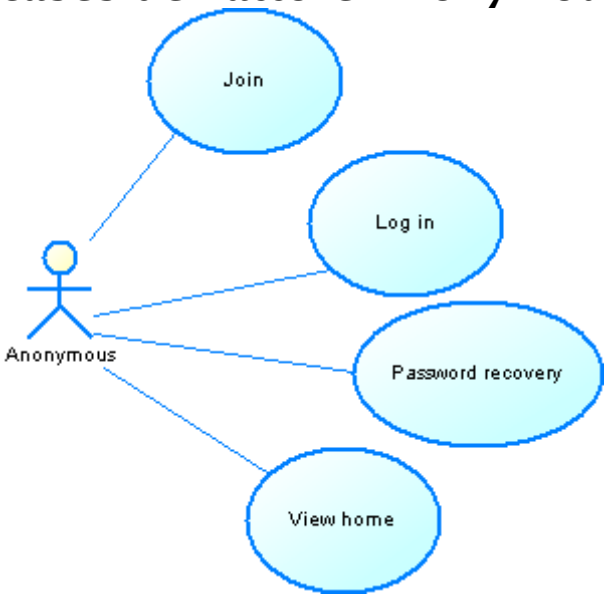

Figura 3.4.: Use case diagram dell'attore Anonymous.

3.7.2.1 Join

Nome	Join
Iniziatore	Anonymous
Scopo	Sottomettere una richiesta di registrazione al sistema
Precondizioni	Il visitatore anonimo non ha mai effettuato una richiesta di registrazione.
Descrizione	1. Il visitatore anonimo chiede di essere registrato 2. Il sistema mostra al visitaore anonimo la pagina contenente il form da compilare 3. Il visitatore anonimo compila il form immettendo i propri dati personali e scegliendo uno username e una password 4. Il visitatore trasmette i dati 5. Il sistema verifica la correttezza dei dati 6. Il sistema memorizza i dati 7. Il sistema notifica la nuova registrazione all'amministratore tramite email 8. Il sistema notifica l'avvenuta registrazione al visitatore.
Alternative	1. Il sistema si ravvede della presenza di campi

	obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form al visitatore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema determina che già esiste un utente registrato con il medesimo username e ripresenta il form al visitatore includendo un messaggio esplicativo del problema incontrato 3. Il sistema determina che già esiste un utente registrato con la medesima email e ripresenta il form al visitatore includendo un messaggio esplicativo del problema incontrato
Postcondizioni	Viene creato un account d'utente per il visitatore contraddistinto dallo stato di “waiting”, cioè in attesa dell'attivazione da parte di un amministratore.

3.7.2.2 Log in

Nome	Log in
Iniziatore	Anonymous
Scopo	Autenticarsi presso il sistema
Precondizioni	Il visitatore anonimo è in possesso di username e password di un utente registrato
Descrizione	1. Il visitatore anonimo chiede di essere autenticato 2. Il sistema mostra al visitatore anonimo la pagina contenente il form per il log in 3. Il visitatore anonimo compila il form immettendo i propri username e password 4. Il visitatore trasmette i dati 5. Il sistema verifica la corrispondenza dei dati inseriti a quelli archiviati 6. Il sistema notifica all'utente l'avvenuto accesso al sistema e espone tutte le funzionalità che un utente può utilizzare
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il

	form al visitatore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema trova corrispondenza in dati relativi ad un utente in attesa o interdetto e mostra all'utente un messaggio in cui lo avvisa dell'impossibilità di accedere 3. Il sistema non trova una corrispondenza in archivio e chiede all'utente di inserire nuovamente i dati. Se è la terza volta consecutiva che l'utente tenta l'accesso con dati errati, il sistema blocca l'accesso al form
Postcondizioni	Il sistema riconosce l'utente e abilita l'accesso alle funzionalità dispobibili

3.7.2.3 Password recovery

Nome	Password recovery
Iniziatore	Anonymous/User/Administrator
Scopo	Recupero password smarrita
Precondizioni	Il visitatore è in possesso di username ma ha dimenticato la propria password
Descrizione	1. Il visitatore chiede di recuperare la password smarrita 2. Il sistema mostra al visitatore la pagina contenente il form per il recupero password 3. Il visitatore anonimo compila il form immettendo il proprio username 4. Il visitatore trasmette i dati 5. Il sistema verifica la presenza dello username immesso tra i dati archiviati 6. Il sistema invia una e-mail contenente la password all'indirizzo di posta dell'utente registrato con quello username 7. Il sistema notifica all'utente l'avvenuto invio della password
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form al visitatore includendo messaggi esplicativi dei problemi incontrati

	2. Il sistema non trova una corrispondenza in archivio e chiede all'utente di inserire nuovamente i dati
Postcondizioni	Una e-mail è stata inviata all'utente

3.7.2.4 View home

Nome	View Home
Iniziatore	Anonymous/User/Administrator
Scopo	Visualizzare la home page dell'applicativo
Precondizioni	Nessuna precondizione
Descrizione	1. Il visitatore accede all'applicazione 2. Il sistema mostra al visitatore la home page contenente i messaggi e le notizie immesse dagli amministratori
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.3 Use cases dell'attore User

L'attore User, cui faremo riferimento anche come utente registrato o semplicemente come utente, concentra attorno a se un numero maggiore di funzionalità. Per questo motivo abbiamo organizzato i diagrammi dei casi d'uso ad esso relativi gerarchicamente. Sarà presentato dapprima un diagramma contenente casi d'uso ad un livello di astrazione più elevato; a seguire saranno aggiunti maggiori particolari.

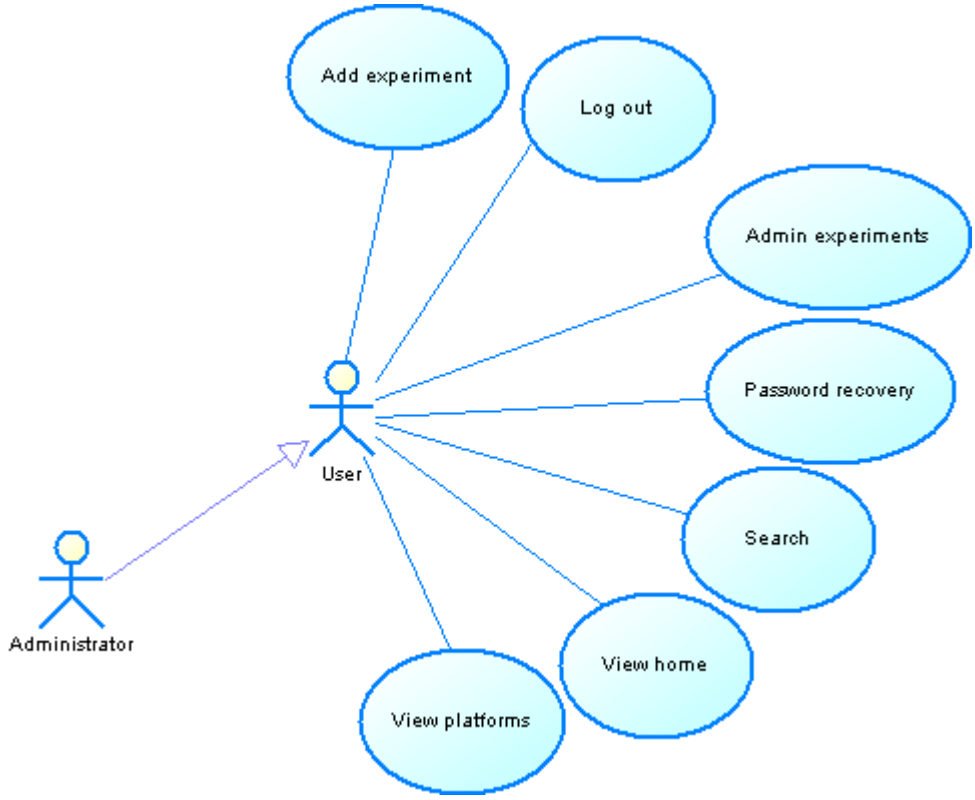

Figura 3.5.: Diagramma generale dei casi d'uso per l'attore User.

Nella lettura di quanto segue, si tenga sempre presente che l'amministratore, essendo una specializzazione dell'attore utente, ne eredita tutti i casi d'uso: tutte le funzionalità offerte all'utente sono anche funzionalità offerte all'amministratore.

3.7.3.1 Add experiment

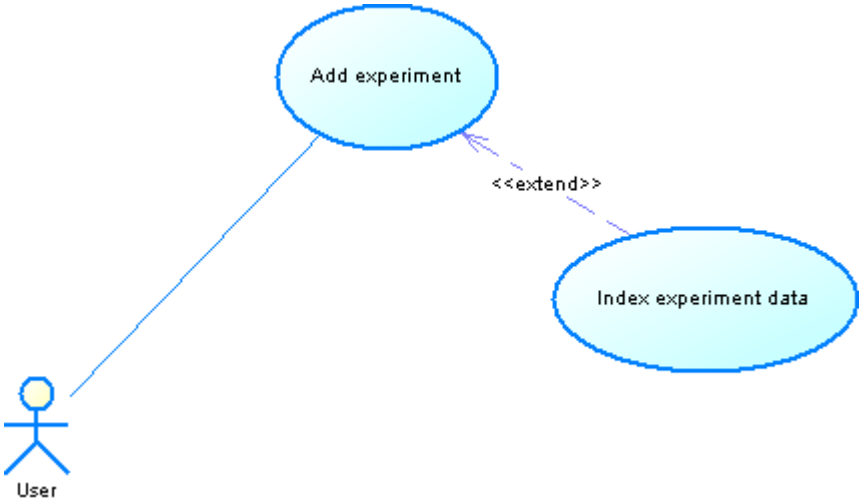

Figura 3.6.: Use case "add experiment" dell'attore user.

Nome	Add experiment
Iniziatore	User/Administrator
Scopo	Aggiunge un esperimento all'archivio
Precondizioni	L'utente ha già raccolto tutti i dati relativi all'esperimento sul proprio PC, comprimendo i dati grezzi e l'eventuale allegato generico.
Descrizione	1. L'utente chiede di inserire un esperimento 2. Il sistema mostra all'utente la prima pagina del wizard per l'inserimento degli esperimenti 3. L'utente compila i form che il sistema gli presenta immettendo i dati dell'esperimento 4. L'utente sceglie se indicizzare subito i dati o affidarsi alla procedura batch di indicizzazione 5. Il sistema notifica all'utente l'avvenuto inserimento dell'esperimento
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form al visitatore includendo messaggi esplicativi dei problemi incontrati

	2. Il sistema trova tra i dati archiviati un esperimento con lo stesso nome e chiede di inserire un nome diverso 3. Il sistema non riesce ad indicizzare i dati (dati normalizzati e liste dei geni regolati) a causa di un errore di formato e notifica l'evento all'utente
Postcondizioni	Un esperimento è memorizzato con lo stato di indicizzato o di non indicizzato a seconda che l'utente abbia scelto o meno l'indicizzazione diretta.

3.7.3.2 Index experiment data

Nome	Index experiment data
Iniziatore	User/Administrator
Scopo	Indicizza i dati di un esperimento
Precondizioni	I dati dell'esperimento sono già stati archiviati dal sistema.
Descrizione	1. L'utente chiede di indicizzare i dati di un esperimento 2. Il sistema indicizza i dati dell'esperimento 3. Il sistema notifica all'utente l'avvenuta indicizzazione dell'esperimento
Alternative	1. Il sistema non riesce ad indicizzare i dati a causa di un errore nel formato e notifica il problema incontrato all'utente
Postcondizioni	L' esperimento è memorizzato con lo stato di indicizzato e i dati sono disponibili per la ricerca

3.7.3.3 Log out

Nome	Log out
Iniziatore	User/Administrator
Scopo	L'utente esce dal sistema
Precondizioni	L'utente è loggato al sistema, cioè ne è stata autenticata l'identità.
Descrizione	1. L'utente chiede di effettuare il log out 2. Il sistema esegue il log out rilasciando le ri-

	sorse dedicate all'utente 3. Il sistema notifica l'avvenuto log out
Alternative	Nessuna
Postcondizioni	L'utente è ora è identificato dal sistema come semplice visistatore

3.7.3.4 View user account

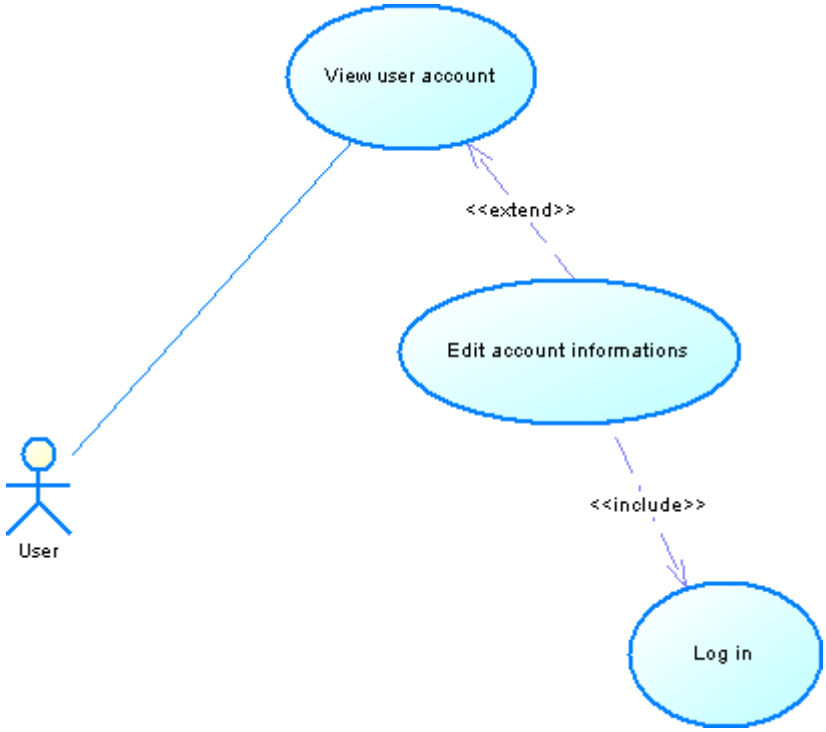

Figura 3.7.: Use case "view user account" dell'attore User.

Nome	View user account
Iniziatore	User/Administrator
Scopo	Visualizza le informazioni personali dell'utente
Precondizioni	L'utente è correttamente loggato al sistema
Descrizione	1. L'utente chiede di visualizzare i propri dati personali 2. Il sistema raccoglie dall'archivio i dati personali dell'utente 3. Il sistema mostra una pagina contenente i dati personali dell'utente

Alternative	Nessuna
Postcondizioni	Nessuna

3.7.3.5 Edit account informations

Nome	Edit account informations
Iniziatore	User/Administrator
Scopo	Modificare le informazioni personali dell'utente
Precondizioni	L'utente è correttamente loggato al sistema
Descrizione	1. L'utente chiede di modificare i propri dati personali 2. Il sistema raccoglie dall'archivio i dati personali dell'utente 3. Il sistema mostra una pagina contenente un form precompilato con i dati personali dell'utente 4. L'utente modifica i propri dati 5. L'utente inserisce i dati modificati 6. Il sistema archivia i nuovi dati sostituendoli a quelli vecchi 7. Il sistema fa il log in dell'utente con i nuovi dati
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form al visitatore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema trova tra i dati archiviati un utente con la stessa e-mail e chiede di inserirne una diversa
Postcondizioni	L'utente è loggato con i nuovi dati personali

3.7.3.6 Password recovery

Nome	Password recovery
Iniziatore	User/Administrator
Scopo	Recupero password smarrita
Precondizioni	L'utente è in possesso di username ma ha dimenti-

	cato la propria password
Descrizione	1. L'utente chiede di recuperare la password smarrita 2. Il sistema mostra all'utente la pagina contenente il form per il recupero password 3. L'utente compila il form immettendo il proprio username 4. L'utente trasmette i dati 5. Il sistema verifica la presenza dello username immesso tra i dati archiviati 6. Il sistema invia una e-mail contenente la password all'indirizzo di posta dell'utente registrato con lo username scelto 7. Il sistema notifica all'utente l'avvenuto invio della password
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form al visitatore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema non trova una corrispondenza in archivio e chiede all'utente di inserire nuovamente i dati
Postcondizioni	Una e-mail, contenente la password, è stata inviata all'utente

3.7.3.7 Search

Nome	Search
Iniziatore	User/Administrator
Scopo	Ricerca tra i dati degli esperimenti
Precondizioni	Nessuna
Descrizione	1. L'utente chiede di effettuare una ricerca 2. Il sistema mostra all'utente la pagina contenente il form per la ricerca 3. L'utente inserisce il testo da ricercare e definisce l'ambito della ricerca 4. L'utente trasmette i dati 5. Il sistema effettua una ricerca sull'indice dei dati

	6. Il sistema presenta all'utente i risultati della ricerca
Alternative	1. Il sistema riconosce un testo illegale e chiede all'utente di effettuare una ricerca con un nuovo testo 2. Il sistema non trova una corrispondenza nell'indice e notifica all'utente l'assenza di risultati della ricerca
Postcondizioni	Nessuna

3.7.3.8 View home

Nome	View Home
Iniziatore	User/Administrator
Scopo	Visualizzare la home page dell'applicativo
Precondizioni	Nessuna
Descrizione	1. L'utente chiede di visualizzare la home page 2. Il sistema mostra all'utente la home page contenente i messaggi e le notizie immesse dagli amministratori
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.3.9 View platforms

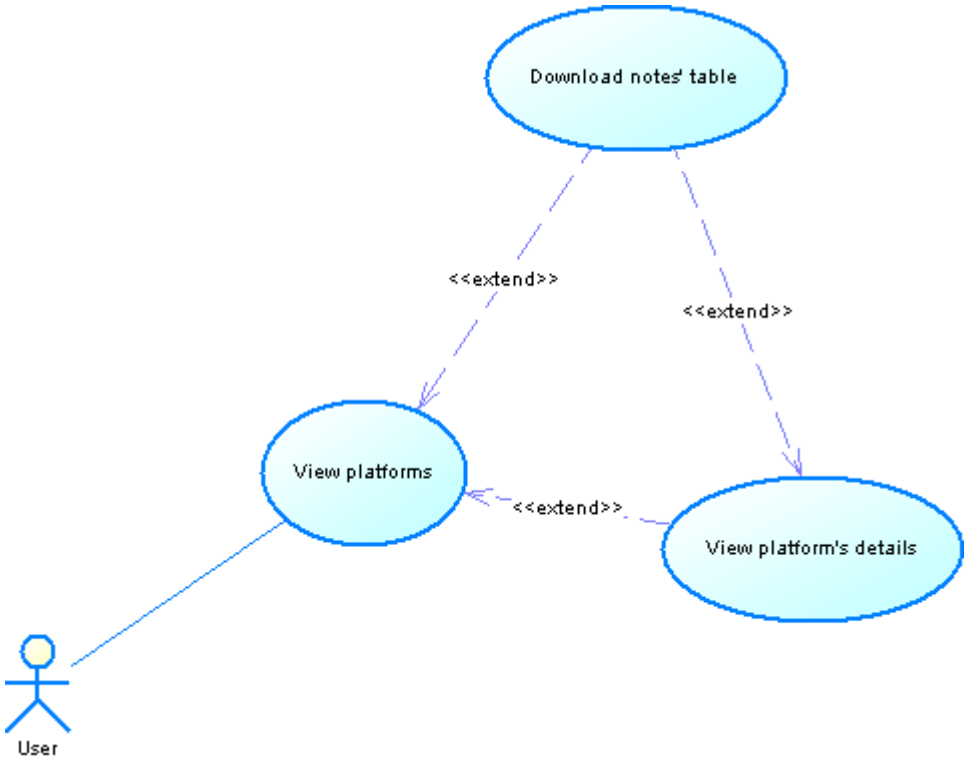

Figura 3.8.: Use case diagram "view platforms" dell'attore user.

Nome	View platforms
Iniziatore	User/Administrator
Scopo	Visualizzare le piattaforme archiviate dall'applicativo
Precondizioni	Nessuna
Descrizione	1. L'utente chiede di visualizzare le piattaforme archiviate 2. Il sistema raccoglie i dati riguardanti le piattaforme archiviate 3. Il sistema mostra all'utente una tabella, eventualmente distribuita su più pagine, contenente le piattaforme archiviate
Alternative	1. Il sistema non trova piattaforme in archivio e mostra un messaggio di notifica all'utente
Postcondizioni	Nessuna

3.7.3.10 View platform's details

Nome	View platform's details
Iniziatore	User/Administrator
Scopo	Visualizzare i dettagli relativi ad una piattaforma
Precondizioni	Il sistema ha mostrato all'utente l'elenco delle piattaforme archiviate
Descrizione	1. L'utente chiede di visualizzare i dettagli di una piattaforma 2. Il sistema raccoglie i dati archiviati riguardanti la piattaforma prescelta 3. Il sistema mostra all'utente tutte le informazioni disponibili sulla piattaforma
Alternative	1. L'utente sceglie di scaricare la tabella delle annotazioni della piattaforma di cui sta visualizzando i dettagli
Postcondizioni	Nessuna

3.7.3.11 Download note's table

Nome	Download note's table
Iniziatore	User/Administrator
Scopo	Scarica la tabella delle annotazioni di una piattaforma
Precondizioni	Il sistema ha mostrato all'utente una pagina contenente i dettagli della piattaforma o l'elenco di tutte le piattaforme archiviate
Descrizione	1. L'utente chiede di scaricare la tabella delle annotazioni di una piattaforma 2. Il sistema preleva la tabella delle annotazioni dall'archivio 3. Il sistema invia all'utente il file contenente i dati 4. L'utente salva i dati scaricati sul proprio PC
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.3.12 Experiments administration

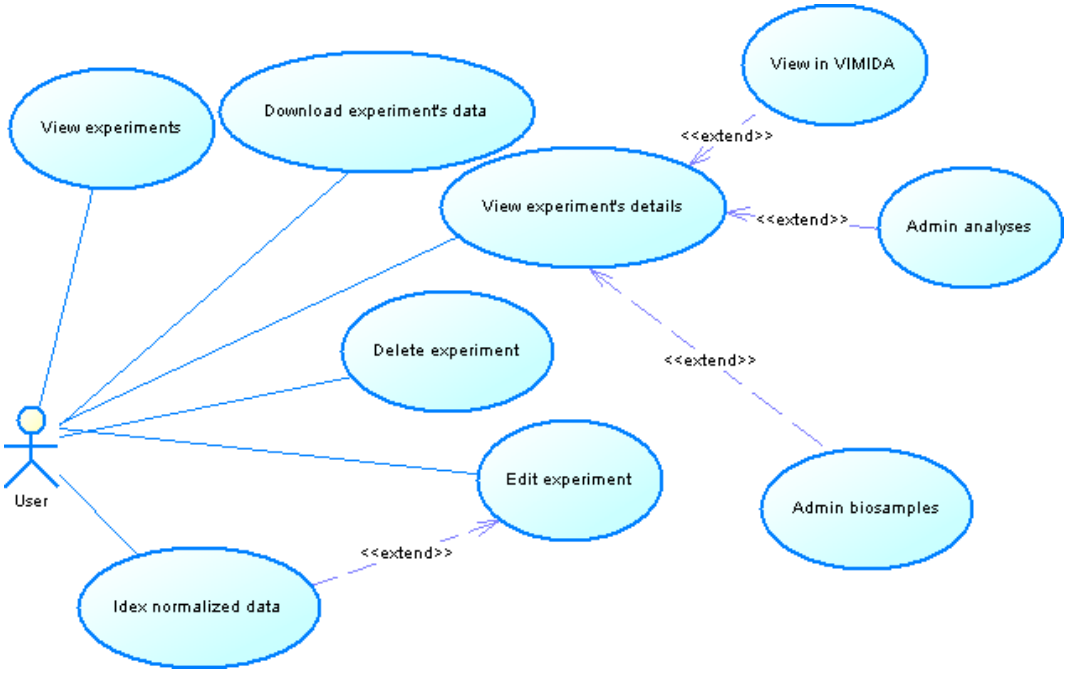

Figura 3.9.: Visione di dettaglio dello use case "Experiments administration".

3.7.3.13 View experiments

Nome	View experiments
Iniziatore	User/Administrator
Scopo	Visualizzare gli esperimenti archiviati dall'applicativo nell'ambito di visibilità dell'utente
Precondizioni	Nessuna
Descrizione	1. L'utente chiede di visualizzare gli esperimenti archiviati 2. Il sistema raccoglie i dati riguardanti gli esperimenti archiviati che l'utente può visualizzare 3. Il sistema mostra all'utente una tabella, eventualmente distribuita su più pagine, contenente gli esperimenti visibili
Alternative	1. Il sistema non trova esprimenti visibili all'utente e mostra un messaggio di notifica
Postcondizioni	Nessuna

3.7.3.14 View experiment's details

Nome	View experiment's details
Iniziatore	User/Administrator
Scopo	Visualizzare i dettagli relativi ad un esperimento
Precondizioni	Il sistema ha mostrato all'utente l'elenco degli esperimenti ad esso visibili
Descrizione	1. L'utente chiede di visualizzare i dettagli di un esperimento 2. Il sistema raccoglie i dati archiviati riguardanti l'esperimento prescelto 3. Il sistema mostra all'utente tutte le informazioni disponibili sull'esperimento
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.3.15 Edit experiment

Nome	Edit experiment
Iniziatore	User/Administrator
Scopo	Modificare i dati relativi ad un esperimento
Precondizioni	Il sistema ha mostrato all'utente l'elenco degli esperimenti ad esso visibili e l'utente è proprietario dell'esperimento prescelto
Descrizione	1. L'utente chiede di modificare i dati di un esperimento 2. Il sistema raccoglie i dati archiviati riguardanti l'esperimento prescelto 3. Il sistema mostra all'utente un form precompilato con i dati correnti dell'esperimento 4. L'utente apporta modifiche ai dati dell'esperimento 5. L'utente inserisce i nuovi dati 6. Il sistema notifica all'utente l'avvenuta modifica dei dati dell'esperimento
Alternative	1. Qualora l'utente abbia scelto di ricaricare i dati normalizzati deve anche scegliere se indicizzarli subito o no

	2. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'utente includendo messaggi esplicativi dei problemi incontrati 3. Il sistema rileva la presenza in archivio di un esperimento con lo stesso nome e chiede che sia modificato 4. Il sistema incontra un problema di indicizzazione a causa di un errore nel formato dei dati normalizzati e notifica l'evento all'utente
Postcondizioni	L'esperimento modificato è memorizzato con lo stato di indicizzato, se è stata scelta l'indicizzazione diretta, o di non indicizzato, se è stata scelta l'indicizzazione differita

3.7.3.16 Delete experiment

Nome	Delete experiment
Iniziatore	User/Administrator
Scopo	Eliminare i dati relativi ad un esperimento
Precondizioni	Il sistema ha mostrato all'utente l'elenco degli esperimenti ad esso visibili e l'utente è proprietario dell'esperimento prescelto
Descrizione	1. L'utente chiede di eliminare un esperimento 2. Il sistema raccoglie i dati archiviati riguardanti l'esperimento prescelto 3. Il sistema mostra all'utente i dati correnti dell'esperimento e chiede conferma dell'eliminazione 4. L'utente sceglie di eliminare l'esperimento 5. Il sistema elimina l'esperimento e i campioni e le analisi collegate 6. Il sistema notifica all'utente l'avvenuta eliminazione dei dati dell'esperimento
Alternative	1. L'utente rinuncia all'eliminazione dell'esperimento e il sistema ripresenta all'utente la pagina di visualizzazione degli esperimenti
Postcondizioni	Nessuna

3.7.3.17 Index normalized data

Nome	Index normalized data
Iniziatore	User/Administrator
Scopo	Indicizzare i dati normalizzati di un esperimento
Precondizioni	I dati normalizzati dell'esperimento non sono stati indicizzati e l'utente è proprietario dell'esperimento stesso
Descrizione	1. L'utente chiede di indicizzare i dati normalizzati di un esperimento 2. Il sistema raccoglie i dati archiviati con l'esperimento prescelto 3. Il sistema indicizza i dati 4. Il sistema notifica all'utente l'avvenuta indicizzazione dei dati dell'esperimento
Alternative	1. Il sistema notifica all'utente l'impossibilità di indicizzare i dati perché non conformi al formato utilizzato dall'applicazione
Postcondizioni	Nessuna

3.7.3.18 View with VIMIDA

Nome	View with VIMIDA
Iniziatore	User/Administrator
Scopo	Visualizzare i dati normalizzati di un esperimento tramite l'applet VIMIDA
Precondizioni	Il sistema ha mostrato all'utente l'elenco degli esperimenti ad esso visibili o i dettagli di un esperimento
Descrizione	1. L'utente chiede di visualizzare i dati normalizzati con VIMIDA 2. Il sistema raccoglie i dati che l'utente ha scelto di visualizzare 3. Il sistema indirizza l'utente alla pagina contenente l'applet di visualizzazione 4. L'utente procede nell'utilizzazione dell'applet
Alternative	Nessuna

Postcondizioni	Nessuna

3.7.3.19 Download experiment's data

Nome	Download experiment's data
Iniziatore	User/Administrator
Scopo	Scaricare i dati archiviati con l'esperimento
Precondizioni	Il sistema ha mostrato all'utente l'elenco degli esperimenti ad esso visibili o i dettagli di un esperimento
Descrizione	1. L'utente chiede di scaricare i dati di un esperimento (dati normalizzati o dati grezzi o allegato) 2. Il sistema raccoglie i dati che l'utente ha scelto di scaricare e li passa al browser 3. L'utente salva sul proprio PC i dati scaricati
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.3.20 Analyses administration

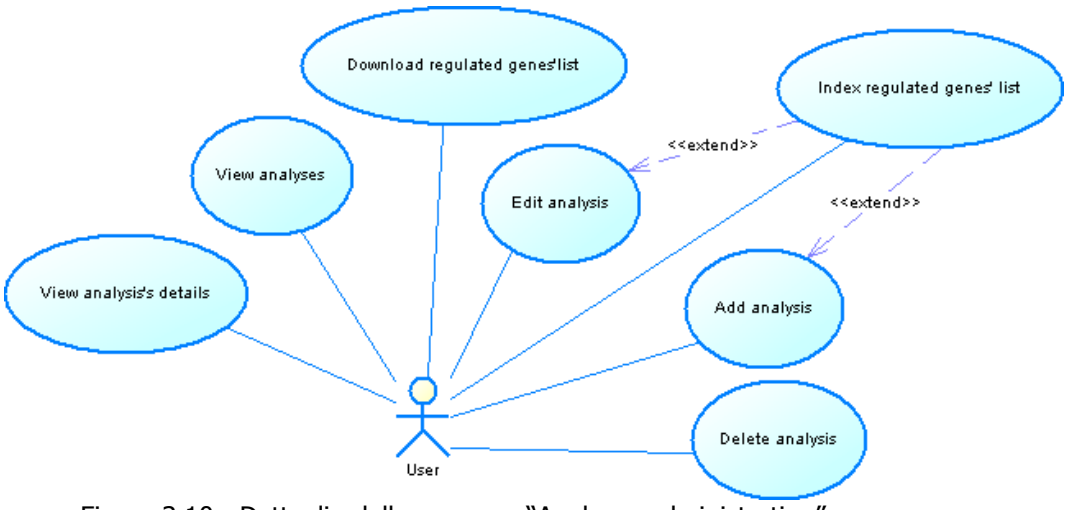

Figura 3.10.: Dettaglio dello use case "Analyses administration".

3.7.3.21 View analyses

Nome	View analyses
Iniziatore	User/Administrator
Scopo	Visualizzare le analisi associate ad un esperimento archiviato
Precondizioni	L'esperimento cui appartengono le analisi è visibile all'utente
Descrizione	1. L'utente chiede di visualizzare le analisi archiviate con un esperimento 2. Il sistema raccoglie i dati sulle analisi archiviate 3. Il sistema mostra all'utente una tabella, eventualmente distribuita su più pagine, contenente le analisi
Alternative	1. Il sistema non trova analisi associate all'esperimento e invia un messaggio di notifica all'utente
Postcondizioni	Nessuna

3.7.3.22 View analysis'details

Nome	View analysis' details
Iniziatore	User/Administrator

Scopo	Visualizzare i dettagli relativi ad una analisi svolta
Precondizioni	L'esperimento cui sono associate le analisi è nell'ambito di visibilità dell'utente
Descrizione	1. L'utente chiede di visualizzare i dettagli riguardo ad una particolare analisi 2. Il sistema raccoglie i dati sull'analisi 3. Il sistema mostra all'utente una pagina contenente i dati raccolti sull'analisi
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.3.23 Add analysis

Nome	Add analysis
Iniziatore	User/Administrator
Scopo	Aggiungere ad un esperimento i dati su una nuova analisi
Precondizioni	L'utente deve essere proprietario dell'esperimento cui associare l'analisi
Descrizione	1. L'utente chiede di inserire una nuova analisi 2. Il sistema mostra all'utente una pagina contenente i form da compilare 3. L'utente compila i form presentati con i dati dell'analisi 4. L'utente inserisce i dati e sceglie se indicizzare subito la lista dei geni regolati 5. Il sistema archivia la nuova analisi 6. Il sistema presenta all'utente un messaggio di notifica dell'avvenuto inserimento
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form al visitatore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema incontra un problema di indicizzazione (se questa è stata richiesta) a causa di un errato formato dei dati e notifica all'utente l'evento

Postcondizioni	Una analisi viene memorizzata in archivio con lo stato di indicizzata, se l'utente ne ha chiesto l'indicizzazione diretta, o non indicizzata, se l'utente ha scelto l'indicizzazione batch differita

3.7.3.24 Edit analysis

Nome	Edit analysis
Iniziatore	User/Administrator
Scopo	Modificare i dati di una analisi
Precondizioni	L'utente deve essere proprietario dell'esperimento cui è associata l'analisi
Descrizione	1. L'utente chiede di modificare una analisi precedentemente archiviata 2. Il sistema raccoglie i dati relativi all'analisi da modificare 3. Il sistema presenta all'utente un form precompilato con i dati relativi all'analisi prescelta 4. L'utente modifica i dati 5. Il sistema archivia l'analisi modificata 6. Il sistema presenta all'utente un messaggio di notifica dell'avvenuta modifica
Alternative	1. Qualora l'utente abbia scelto di ricaricare la lista dei geni regolati deve scegliere anche se indicizzare subito i nuovi dati 2. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form al visitatore includendo messaggi esplicativi dei problemi incontrati 3. Il sistema incontra un problema di indicizzazione (se questa è stata richiesta) a causa di un errato formato dei dati e notifica all'utente l'evento
Postcondizioni	L'analisi modificata viene memorizzata in archivio con lo stato di indicizzata, se l'utente ne ha chiesto l'indicizzazione diretta, o non indicizzata, se l'utente ha scelto l'indicizzazione batch differita

3.7.3.25 Delete analysis

Nome	Delete analysis
Iniziatore	User/Administrator
Scopo	Eliminare una analisi
Precondizioni	L'utente deve essere proprietario dell'esperimento cui è associata l'analisi
Descrizione	1. L'utente chiede di eliminare una analisi precedentemente archiviata nel sistema 2. Il sistema raccoglie i dati relativi all'analisi da eliminare 3. Il sistema presenta all'utente i dati dell'analisi da eliminare e chiede conferma dell'eliminazione 4. L'utente conferma l'eliminazione 5. Il sistema elimina l'analisi 6. Il sistema presenta all'utente un messaggio di notifica dell'avvenuta eliminazione
Alternative	1. L'utente rinuncia all'eliminazione dell'analisi e il sistema presenta di nuovo all'utente una lista delle analisi associate all'esperimento
Postcondizioni	Nessuna

3.7.3.26 Download regulated genes' list

Nome	Download regulated gene's list
Iniziatore	User/Administrator
Scopo	Scarica la lista di geni regolati di una analisi
Precondizioni	Il sistema ha mostrato all'utente una pagina contenente i dettagli dell'analisi o l'elenco di tutte le analisi archiviate. L'esperimento cui è associata l'analisi rientra nell'ambito di visibilità dell'utente.
Descrizione	1. L'utente chiede di scaricare la lista dei geni regolati di una analisi 2. Il sistema preleva la lista dei geni regolati dall'archivio 3. Il sistema invia all'utente il file contenente i dati 4. L'utente salva i dati scaricati sul proprio PC

Alternative	Nessuna
Postcondizioni	Nessuna

3.7.3.27 Index regulated gene's list

Nome	Index regulated genes' list
Iniziatore	User/Administrator
Scopo	Indicizzare la lista dei geni regolati di una analisi
Precondizioni	La lista dei geni regolati dell'analisi scelta non è stata indicizzata e l'utente è proprietario dell'esperimento cui è associata l'analisi
Descrizione	1. L'utente chiede di indicizzare la lista dei geni regolati di una analisi 2. Il sistema raccoglie i dati archiviati con l'analisi prescelta 3. Il sistema indicizza i dati 4. Il sistema notifica all'utente l'avvenuta indicizzazione dei dati dell'analisi
Alternative	1. Il sistema notifica all'utente l'impossibilità di indicizzare i dati perché non conformi al formato utilizzato dall'applicazione
Postcondizioni	Nessuna

3.7.3.28 Biosamples administration

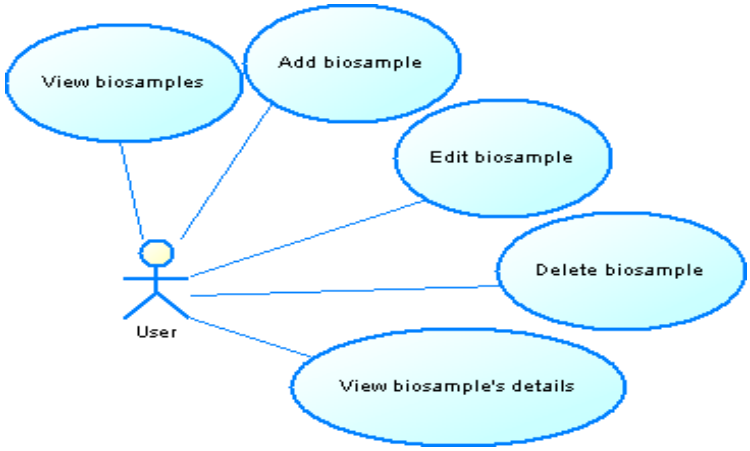

Figura 3.11.: Dettagli dello use case "Biosamples administration".

3.7.3.29 View biosamples

Nome	View biosamples
Iniziatore	User/Administrator
Scopo	Visualizzare i campioni associata ad un esperimento archiviato
Precondizioni	L'esperimento cui appartengono i campioni è visibile all'utente
Descrizione	1. L'utente chiede di visualizzare i campioni archiviate con un esperimento 2. Il sistema raccoglie i dati sui campioni archiviati 3. Il sistema mostra all'utente una tabella, eventualmente distribuita su più pagine, contenente i campioni
Alternative	1. Il sistema non trova campioni associati all'esperimento e invia un messaggio di notifica all'utente
Postcondizioni	Nessuna

3.7.3.30 View biosample's details

Nome	View biosample's details
Iniziatore	User/Administrator
Scopo	Visualizzare i dettagli relativi ad un campione esa-

	minato
Precondizioni	Il sistema ha mostrato all'utente una lista dei campioni associati ad un certo esperimento. L'esperimento cui è associato il campione è nell'ambito di visibilità dell'utente.
Descrizione	1. L'utente chiede di visualizzare i dettagli riguardo ad uno dei campioni analizzati 2. Il sistema raccoglie i dati sul campione 3. Il sistema mostra all'utente una pagina contenente i dati raccolti sul campione
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.3.31 Add biosample

Nome	Add biosample
Iniziatore	User/Administrator
Scopo	Aggiungere ad un esperimento i dati su un nuovo campione
Precondizioni	L'utente deve essere proprietario dell'esperimento cui associare il campione
Descrizione	1. L'utente chiede di inserire un nuovo campione 2. Il sistema mostra all'utente una pagina contenente i form da compilare 3. L'utente compila i form presentati con i dati del nuovo campione 4. L'utente inserisce i dati 5. Il sistema archivia il nuovo campione 6. Il sistema presenta all'utente un messaggio di notifica dell'avvenuto inserimento
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form al visitatore includendo messaggi esplicativi dei problemi incontrati
Postcondizioni	Nessuna

3.7.3.32 Edit biosample

Nome	Edit biosample
Iniziatore	User/Administrator
Scopo	Modificare i dati di una analisi
Precondizioni	Il sistema ha presentato una pagina contenente i campioni di un esperimento. L'utente deve essere proprietario dell'esperimento cui è associato il campione.
Descrizione	1. L'utente chiede di modificare un campione precedentemente archiviato 2. Il sistema raccoglie i dati relativi al campione da modificare 3. Il sistema presenta all'utente un form precompilato con i dati relativi al campione prescelto 4. L'utente modifica i dati 5. Il sistema archivia il campione modificato 6. Il sistema presenta all'utente un messaggio di notifica dell'avvenuta modifica
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form al visitatore includendo messaggi esplicativi dei problemi incontrati
Postcondizioni	Nessuna

3.7.3.33 Delete biosample

Nome	Delete biosample
Iniziatore	User/Administrator
Scopo	Eliminare una analisi
Precondizioni	L'utente deve essere proprietario dell'esperimento cui è associato il campione. Il campione non deve essere stato utilizzato in una analisi.
Descrizione	1. L'utente chiede di eliminare un campione precedentemente archiviato nel sistema 2. Il sistema raccoglie i dati relativi al campione da eliminare

	3. Il sistema presenta all'utente i dati del campione da eliminare e chiede conferma dell'eliminazione 4. L'utente conferma l'eliminazione 5. Il sistema elimina il campione 6. Il sistema presenta all'utente un messaggio di notifica dell'avvenuta eliminazione
Alternative	1. L'utente rinuncia all'eliminazione del campione e il sistema presenta di nuovo all'utente una lista di tutti i campioni associati all'esperimento
Postcondizioni	Nessuna

3.7.4 Use cases dell'attore Administrator

Anche l'attore Administrator, cui faremo riferimento anche come amministratore, concentra attorno a se un numero di funzionalità tale da richiedere una organizzazione gerarchica dei diagrammi dei casi d'uso ad esso relativi. Sarà presentato dapprima un diagramma contenente casi d'uso ad un livello di astrazione più elevato; a seguire saranno aggiunti maggiori particolari e dettagli.

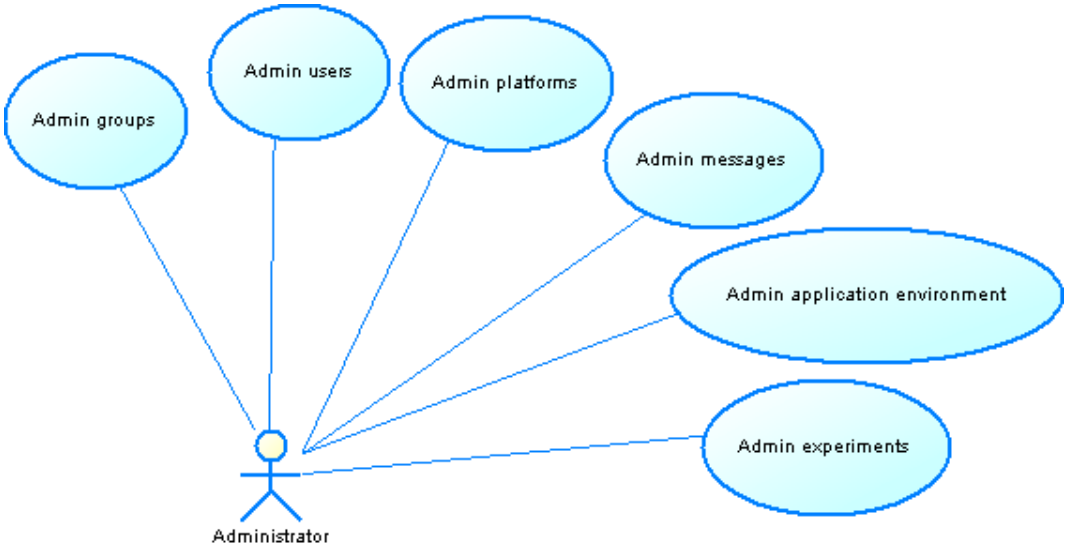

Figura 3.12.: Use cases dell'attore Administrator al più alto livello di astrazione.

Come già detto, si tenga conto del fatto che l'amministratore, essendo una specializzazione dell'attore utente, ne eredita tutti i casi d'uso. Ai casi d'uso ereditati dall'utente, dei quali riporteremo quelli che presentano differenze rispetto a quanto già illustrato, si aggiungono quelli peculiari dell'amministratore del sistema.

3.7.4.1 Groups administration

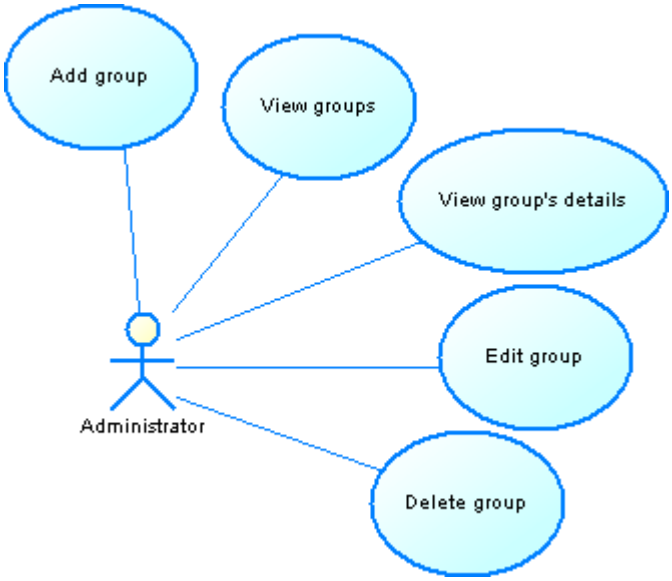

Figura 3.13.: Use case diagram "Groups administration" dell'attore Administrator.

3.7.4.2 View groups

Nome	View groups
Iniziatore	Administrator
Scopo	Visualizzare i gruppi di ricerca o gruppi di utenti archiviati
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare i gruppi 2. Il sistema raccoglie i dati riguardanti tutti i gruppi archiviati 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente i gruppi
Alternative	1. Il sistema non trova gruppi in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.3 View group's details

Nome	View group's details
Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad un gruppo
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei gruppi
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un gruppo 2. Il sistema raccoglie i dati archiviati riguardanti il gruppo prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sul gruppo
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.4 Add group

Nome	Add group
Iniziatore	Administrator
Scopo	Inserire un nuovo gruppo
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei gruppi
Descrizione	1. L'amministratore chiede di inserire un nuovo gruppo 2. Il sistema mostra all'amministratore il form da compilare con i dati del gruppo 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo gruppo 5. Il sistema notifica all'amministratore l'avvenuto salvataggio del gruppo
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un gruppo con lo stesso nome e chiede che sia

	scelto un nome differente
Postcondizioni	Nessuna

3.7.4.5 Edit group

Nome	Edit group
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un gruppo
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei gruppi
Descrizione	1. L'amministratore chiede di modificare i dati di un gruppo 2. Il sistema raccoglie i dati archiviati riguardo al gruppo prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti del gruppo 4. L'amministratore apporta modifiche ai dati del gruppo 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate al gruppo 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati del gruppo
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un gruppo con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.6 Delete group

Nome	Delete group
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un gruppo

Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei gruppi
Descrizione	1. L'amministratore chiede di eliminare un gruppo 2. Il sistema raccoglie i dati archiviati riguardo al gruppo prescelto 3. Il sistema mostra all'amministratore i dati correnti del gruppo e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare il gruppo 5. Il sistema elimina il gruppo 6. Il sistema notifica all'amministratore l'avvenuta eliminazione del gruppo
Alternative	1. L'amministratore rinuncia all'eliminazione del gruppo e il sistema ripresenta all'amministratore la pagina di visualizzazione dei gruppi 1. Il sistema rileva la presenza di uno o più utenti associati al gruppo e chiede di eliminare quegli utenti prima del gruppo
Postcondizioni	Nessuna

3.7.4.7 Users administration

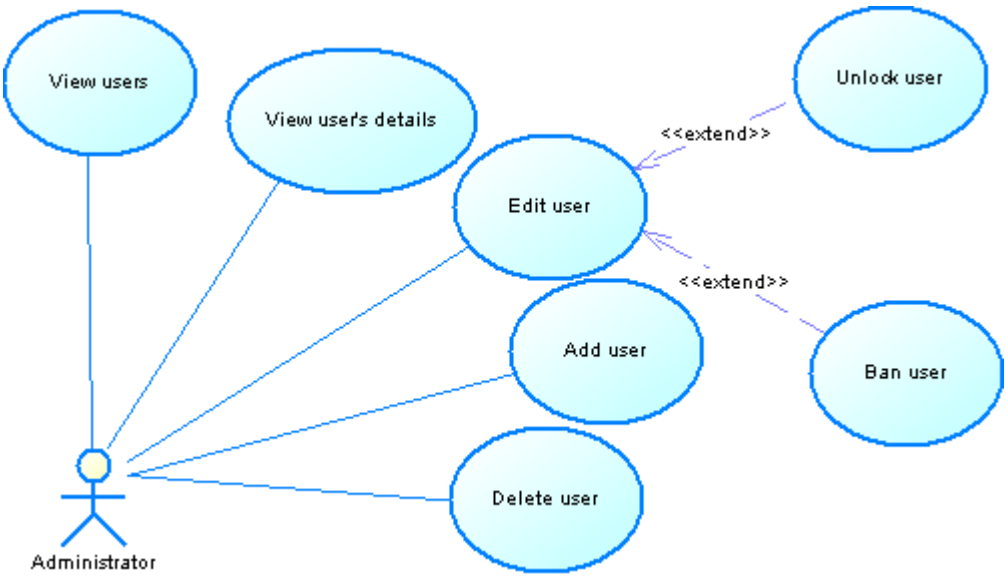

Figura 3.14.: Use case diagram "Users administration" dell'attore Administrator.

3.7.4.8 View users

Nome	View users
Iniziatore	Administrator
Scopo	Visualizzare gli utente del sistema
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare gli utenti del sistema 2. Il sistema raccoglie i dati riguardanti tutti gli utenti registrati 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente gli utenti
Alternative	1. Il sistema non trova utenti in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.9 View user's details

Nome	View user's details
Iniziatore	Administrator

Scopo	Visualizzare i dettagli relativi ad un utente
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco degli utenti
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un utente 2. Il sistema raccoglie i dati archiviati riguardo l'utente prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sull'utente
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.10 Add user

Nome	Add user
Iniziatore	Administrator
Scopo	Inserire un nuovo utente dall'interfaccia di amministrazione
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco degli utenti
Descrizione	1. L'amministratore chiede di inserire un nuovo utente 2. Il sistema mostra all'amministratore il form da compilare con i dati dell'utente 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo utente 5. Il sistema notifica all'amministratore l'avvenuto salvataggio dell'utente
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un utente con lo stesso username e chiede che sia scelto uno username differente 3. Il sistema rileva la presenza in archivio di un utente con la stessa e-mail e chiede che sia scelta una e-mail diversa

Postcondizioni	Nessuna

3.7.4.11 Edit user

Nome	Edit user
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un utente
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco degli utenti
Descrizione	1. L'amministratore chiede di modificare i dati di un utente 2. Il sistema raccoglie i dati archiviati riguardo all'utente prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti dell'utente 4. L'amministratore apporta modifiche ai dati dell'utente 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate all'utente 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati dell'utente
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un utente con lo stesso username e chiede che sia scelto uno username differente 3. Il sistema rileva la presenza in archivio di un utente con la stessa e-mail e chiede che sia scelta una e-mail differente
Postcondizioni	Nessuna

3.7.4.12 Delete user

Nome	Delete user
Iniziatore	Administrator

Scopo	Eliminare i dati relativi ad un utente
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco degli utenti
Descrizione	1. L'amministratore chiede di eliminare un utente 2. Il sistema raccoglie i dati archiviati riguardo all'utente prescelto 3. Il sistema mostra all'amministratore i dati correnti dell'utente e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare il utente 5. Il sistema elimina l'utente 6. Il sistema notifica all'amministratore l'avvenuta eliminazione dell'utente
Alternative	2. Il sistema rileva la presenza di uno o più esperimenti associati all'utente e chiede di eliminare quegli esperimenti prima dell'utente 3. L'amministratore rinuncia all'eliminazione dell'esperimento e il sistema ripresenta all'amministratore la pagina di visualizzazione degli utenti
Postcondizioni	Nessuna

3.7.4.13 Unlock user

Nome	Unlock user
Iniziatore	Administrator
Scopo	Sbloccare l'account di un utente
Precondizioni	Il sistema ha notificato all'amministratore l'iscrizione di un nuovo utente. L'utente è nello stato di waiting e non può effettuare il log in
Descrizione	1. L'amministratore chiede di modificare i dati di un utente 2. Il sistema raccoglie i dati archiviati riguardo all'utente prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti dell'utente

	4. L'amministratore modifica lo stato dell'utente: lo sblocca 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate all'utente 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati dell'utente
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un utente con lo stesso username e chiede che sia scelto uno username differente 3. Il sistema rileva la presenza in archivio di un utente con la stessa e-mail e chiede che sia scelta una e-mail differente
Postcondizioni	Nessuna

3.7.4.14 Ban user

Nome	Ban user
Iniziatore	Administrator
Scopo	Interdire un utente dall'utilizzo del sistema
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco degli utenti
Descrizione	1. L'amministratore chiede di modificare i dati di un utente 2. Il sistema raccoglie i dati archiviati riguardo all'utente prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti dell'utente 4. L'amministratore apporta modifiche allo stato dell'utente: lo interdice 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate all'utente 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati dell'utente

Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un utente con lo stesso username e chiede che sia scelto uno username differente 3. Il sistema rileva la presenza in archivio di un utente con la stessa e-mail e chiede che sia scelta una e-mail differente
Postcondizioni	Nessuna

3.7.4.15 Platforms administration

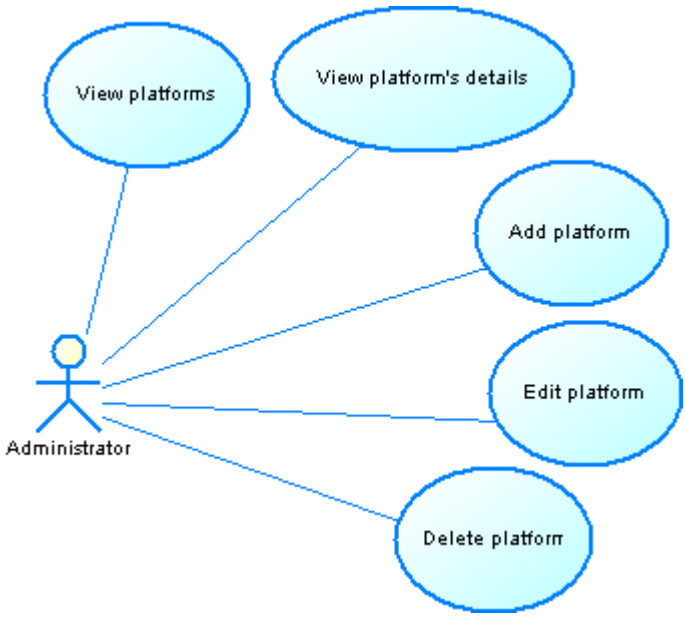

Figura 3.15.: Use case diagram "Platforms administration" dell'attore Administrator.

3.7.4.16 View platforms

Nome	View platforms
Iniziatore	Administrator
Scopo	Visualizzare le piattaforme archiviate nel sistema
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare le piattaforme 2. Il sistema raccoglie i dati riguardanti tutte le piattaforme archiviate 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente le piattaforme
Alternative	1. Il sistema non trova piattaforme in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.17 View platform's details

Nome	View platform's details

Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad una piattaforma
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle piattaforme
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di una piattaforma 2. Il sistema raccoglie i dati archiviati riguardo alla piattaforma prescelta 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sulla piattaforma
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.18 Add platform

Nome	Add platform
Iniziatore	Administrator
Scopo	Inserire una nuova piattaforma
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle piattaforme
Descrizione	1. L'amministratore chiede di inserire una nuova piattaforma 2. Il sistema mostra all'amministratore il form da compilare con i dati della piattaforma 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva la nuova piattaforma 5. Il sistema notifica all'amministratore l'avvenuto salvataggio della piattaforma
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di una piattaforma con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.19 Edit platform

Nome	Edit platform
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad una piattaforma
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle piattaforme
Descrizione	1. L'amministratore chiede di modificare i dati di una piattaforma 2. Il sistema raccoglie i dati archiviati riguardo alla piattaforma prescelta 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti della piattaforma 4. L'amministratore apporta modifiche ai dati della piattaforma 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate alla piattaforma 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati della piattaforma
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di una piattaforma con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.20 Delete platform

Nome	Delete platform
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad una piattaforma
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle piattaforme
Descrizione	1. L'amministratore chiede di eliminare una

	piattaforma 2. Il sistema raccoglie i dati archiviati riguardo alla piattaforma prescelta 3. Il sistema mostra all'amministratore i dati correnti della piattaforma e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare la piattaforma 5. Il sistema elimina la piattaforma 6. Il sistema notifica all'amministratore l'avvenuta eliminazione della piattaforma
Alternative	1. L'amministratore rinuncia all'eliminazione della piattaforma e il sistema ripresenta all'amministratore la pagina di visualizzazione delle piattaforme 1. Il sistema rileva la presenza di uno o più esperimenti associati alla piattaforma e chiede di eliminare quegli esperimenti prima della piattaforma
Postcondizioni	Nessuna

3.7.4.21 Messages administration

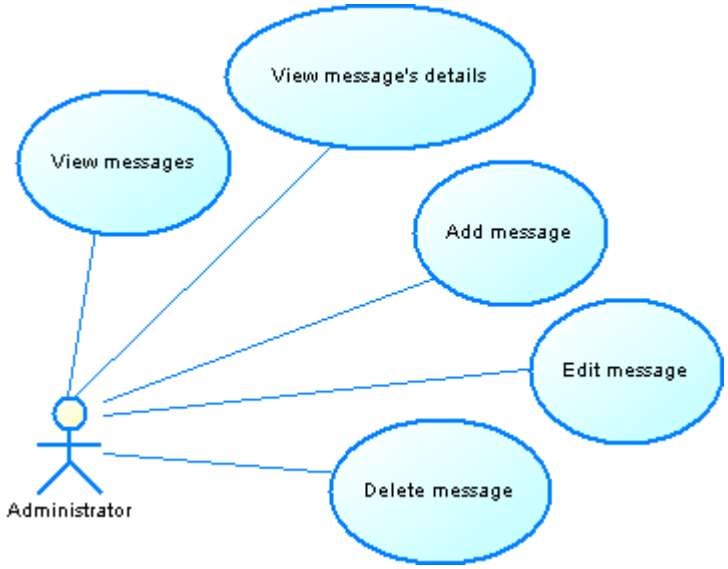

Figura 3.16.: Use case diagram "Messages administration" dell'attore Administrator.

3.7.4.22 View messages

Nome	View messages
Iniziatore	Administrator
Scopo	Visualizzare i messaggi/notizie archiviati dal sistema
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare i messaggi 2. Il sistema raccoglie i dati riguardanti tutti i messaggi archiviati 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente i messaggi
Alternative	1. Il sistema non trova messaggi in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.23 View message's details

Nome	View message's details

Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad un messaggio
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei messaggi
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un messaggio 2. Il sistema raccoglie i dati archiviati riguardo al messaggio prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sul messaggio
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.24 Add message

Nome	Add message
Iniziatore	Administrator
Scopo	Inserire un nuovo messaggio
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei messaggi
Descrizione	1. L'amministratore chiede di inserire un nuovo messaggio 2. Il sistema mostra all'amministratore il form da compilare con i dati del messaggio 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo messaggio 5. Il sistema notifica all'amministratore l'avvenuto salvataggio del messaggio
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati
Postcondizioni	Nessuna

3.7.4.25 Edit message

Nome	Edit message

Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un messaggio
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei messaggi
Descrizione	1. L'amministratore chiede di modificare i dati di un messaggio 2. Il sistema raccoglie i dati archiviati riguardo al messaggio prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti del messaggio 4. L'amministratore apporta modifiche ai dati del messaggio 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate al messaggio 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati del messaggio
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati
Postcondizioni	Nessuna

3.7.4.26 Delete message

Nome	Delete message
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un messaggio
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei messaggi
Descrizione	1. L'amministratore chiede di eliminare un messaggio 2. Il sistema raccoglie i dati archiviati riguardo al messaggio prescelto 3. Il sistema mostra all'amministratore i dati correnti del messaggio e chiede conferma dell'eliminazione

	4. L'amministratore sceglie di eliminare il messaggio 5. Il sistema elimina il messaggio 6. Il sistema notifica all'amministratore l'avvenuta eliminazione del messaggio
Alternative	1. L'amministratore rinuncia all'eliminazione del messaggio e il sistema ripresenta all'amministratore la pagina di visualizzazione dei messaggi
Postcondizioni	Nessuna

3.7.4.27 Application environment's administration

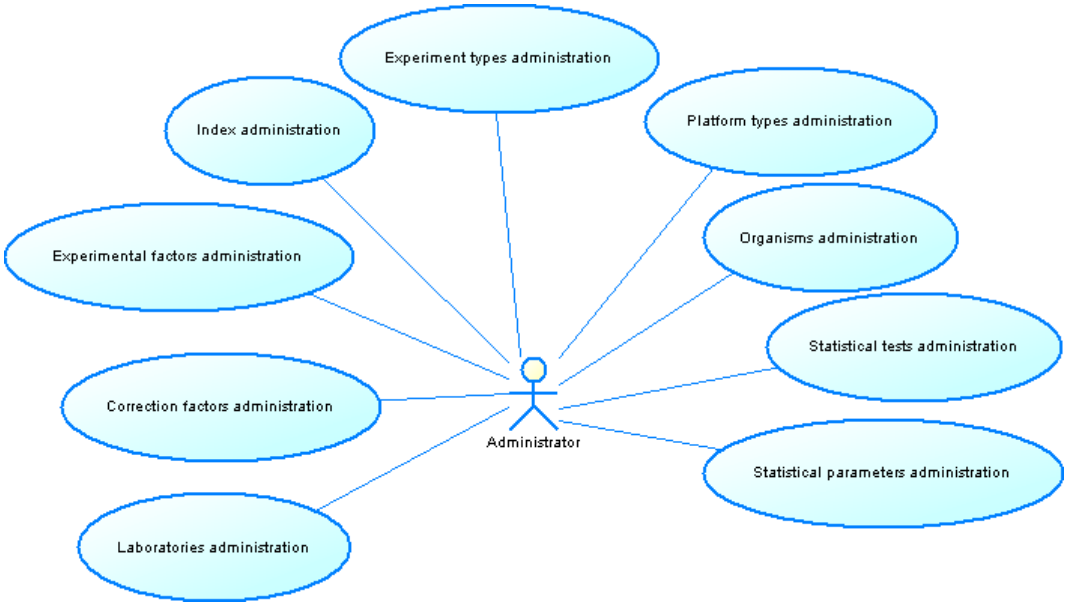

Figura 3.17.: Use case diagram "Application environment's administration" dell'attore Administrator.

3.7.4.28 Laboratories administration

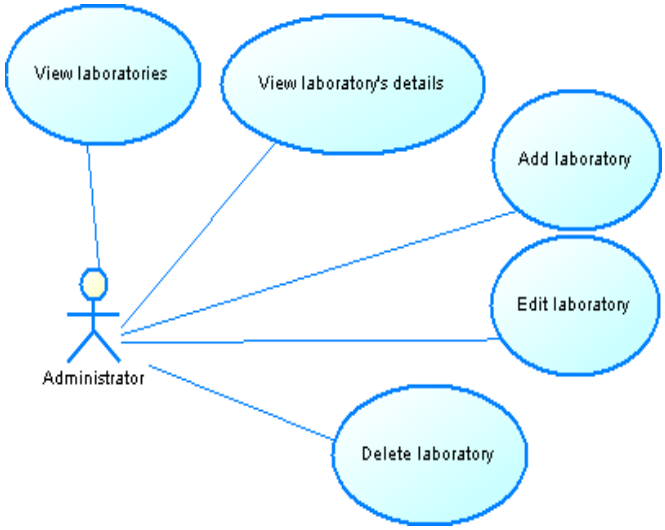

Figura 3.18.: Use case "Laboratories administration" dell'attore Administrator.

3.7.4.29 View laboratories

Nome	View laboratories
Iniziatore	Administrator
Scopo	Visualizzare i laboratori archiviati

Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare i laboratori 2. Il sistema raccoglie i dati riguardanti tutti i laboratori archiviati 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente i laboratori
Alternative	1. Il sistema non trova laboratori in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.30 View laboratory's details

Nome	View laboratory's details
Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad un laboratorio
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei laboratori
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un laboratorio 2. Il sistema raccoglie i dati archiviati riguardo al laboratorio prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sul laboratorio
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.31 Add laboratory

Nome	Add laboratory
Iniziatore	Administrator
Scopo	Inserire un nuovo laboratorio
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei laboratori
Descrizione	1. L'amministratore chiede di inserire un nuovo laboratorio

	2. Il sistema mostra all'amministratore il form da compilare con i dati del laboratorio 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo laboratorio 5. Il sistema notifica all'amministratore l'avvenuto salvataggio del laboratorio
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un laboratorio con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.32 Edit laboratory

Nome	Edit laboratory
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un laboratorio
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei laboratori
Descrizione	1. L'amministratore chiede di modificare i dati di un laboratorio 2. Il sistema raccoglie i dati archiviati riguardo al laboratorio prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti del laboratorio 4. L'amministratore apporta modifiche ai dati del laboratorio 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate al laboratorio 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati del laboratorio
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il

	form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un laboratorio con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.33 Delete laboratory

Nome	Delete laboratory
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un laboratorio
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei laboratori
Descrizione	1. L'amministratore chiede di eliminare un laboratorio 2. Il sistema raccoglie i dati archiviati riguardo al laboratorio prescelto 3. Il sistema mostra all'amministratore i dati correnti del laboratorio e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare il laboratorio 5. Il sistema elimina il laboratorio 6. Il sistema notifica all'amministratore l'avvenuta eliminazione del laboratorio
Alternative	1. L'amministratore rinuncia all'eliminazione del laboratorio e il sistema ripresenta all'amministratore la pagina di visualizzazione dei laboratori 2. Il sistema rileva la presenza di uno o più esperimenti associati al laboratorio e chiede di eliminare quegli esperimenti prima di eliminare il laboratorio
Postcondizioni	Nessuna

3.7.4.34 Correction factors administration

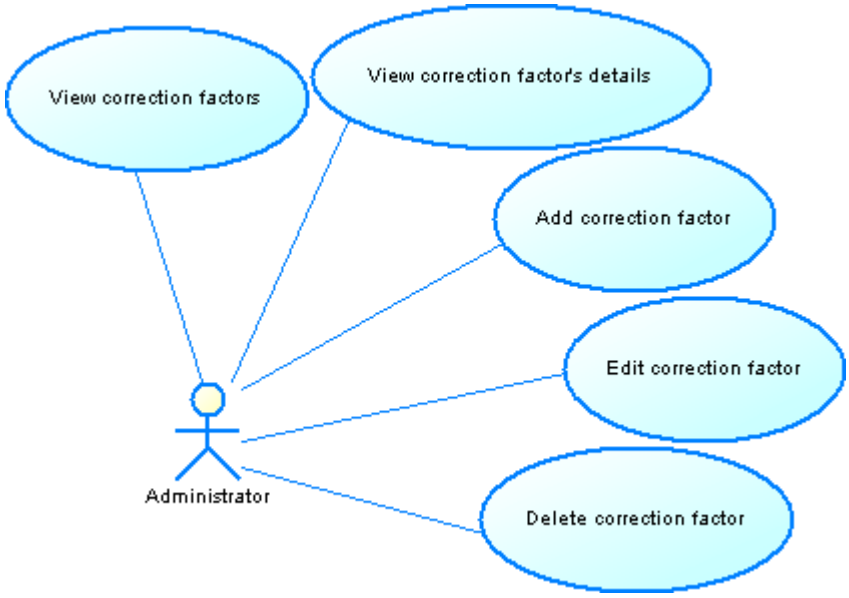

Figura 3.19.: Use case diagram "correction factors administration" dell'attore Administrator.

3.7.4.35 View correction factors

Nome	View correction factors
Iniziatore	Administrator
Scopo	Visualizzare i fattori di correzione archiviati
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare i fattori di correzione 2. Il sistema raccoglie i dati riguardanti tutti i fattori di correzione archiviati 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente i fattori di correzione
Alternative	1. Il sistema non trova fattori di correzione in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.36 View correction factor's details

Nome	View correction factor's details

Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad un fattore di correzione
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei fattori di correzione
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un fattore di correzione 2. Il sistema raccoglie i dati archiviati riguardo al fattore di correzione prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sul fattore di correzione
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.37 Add correction factor

Nome	Add correction factor
Iniziatore	Administrator
Scopo	Inserire un nuovo fattore di correzione
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei fattori di correzione
Descrizione	1. L'amministratore chiede di inserire un nuovo fattore di correzione 2. Il sistema mostra all'amministratore il form da compilare con i dati del fattore di correzione 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo fattore di correzione 5. Il sistema notifica all'amministratore l'avvenuto salvataggio del fattore di correzione
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un

	fattore di correzione con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.38 Edit correction factor

Nome	Edit correction factor
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un fattore di correzione
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei fattori di correzione
Descrizione	1. L'amministratore chiede di modificare i dati di un fattore di correzione 2. Il sistema raccoglie i dati archiviati riguardo al fattore di correzione prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti del fattore di correzione 4. L'amministratore apporta modifiche ai dati del fattore di correzione 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate al fattore di correzione 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati del fattore di correzione
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un fattore di correzione con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.39 Delete correction factor

Nome	Delete correction factor

Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un fattore di correzione
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei fattori di correzione
Descrizione	1. L'amministratore chiede di eliminare un fattore di correzione 2. Il sistema raccoglie i dati archiviati riguardo al fattore di correzione prescelto 3. Il sistema mostra all'amministratore i dati correnti del fattore di correzione e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare il fattore di correzione 5. Il sistema elimina il fattore di correzione 6. Il sistema notifica all'amministratore l'avvenuta eliminazione del fattore di correzione
Alternative	1. L'amministratore rinuncia all'eliminazione del fattore di correzione e il sistema ripresenta all'amministratore la pagina di visualizzazione dei fattori di correzione. 2. Il sistema rileva la presenza di uno o più esperimenti associati al fattore di correzione e chiede di eliminare quegli esperimenti prima di eliminare il fattore di correzione
Postcondizioni	Nessuna

3.7.4.40 Experimental factors administration

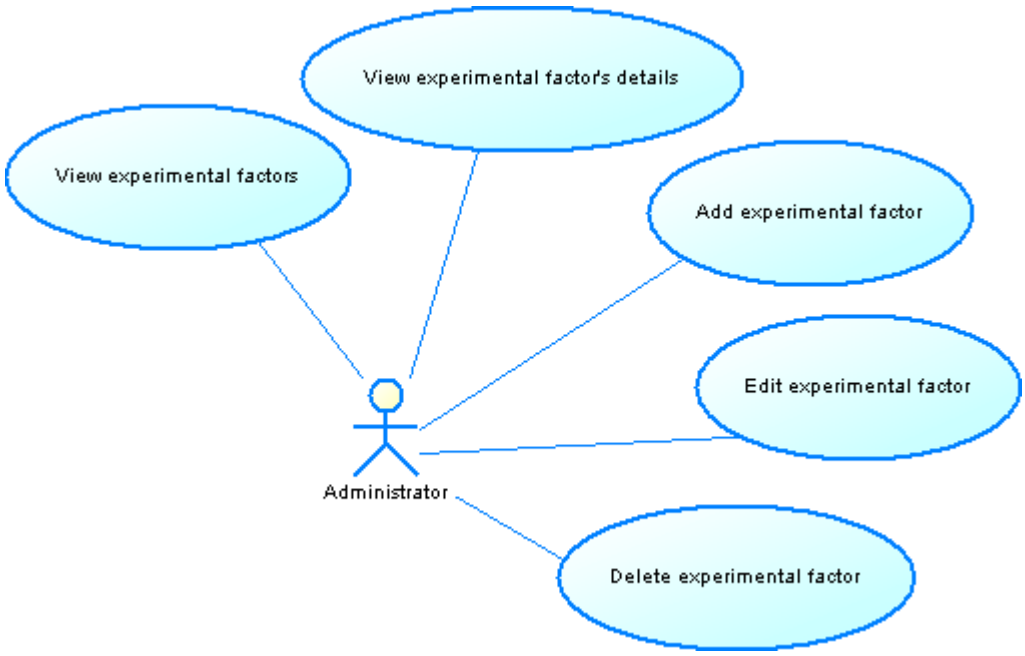

Figura 3.20.: Use case diagram "experimental factors administration" dell'attore Administrator.

3.7.4.41 View experimental factors

Nome	View experimental factors
Iniziatore	Administrator
Scopo	Visualizzare i fattori sperimentali archiviati
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare i fattori sperimentali 2. Il sistema raccoglie i dati riguardanti tutti i fattori sperimentali archiviati 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente i fattori sperimentali
Alternative	1. Il sistema non trova fattori sperimentali in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.42 View experimental factor's details

Nome	View experimental factor's details
Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad un fattore sperimentale
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei fattori sperimentali
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un fattore sperimentale 2. Il sistema raccoglie i dati archiviati riguardo al fattore sperimentale prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sul fattore sperimentale
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.43 Add experimental factor

Nome	Add experimental factor
Iniziatore	Administrator
Scopo	Inserire un nuovo fattore sperimentale
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei fattori sperimentali
Descrizione	1. L'amministratore chiede di inserire un nuovo fattore sperimentale 2. Il sistema mostra all'amministratore il form da compilare con i dati del fattore sperimentale 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo fattore sperimentale 5. Il sistema notifica all'amministratore l'avvenuto salvataggio del fattore sperimentale
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messag-

	gi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un fattore sperimentale con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.44 Edit experimental factor

Nome	Edit experimental factor
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un fattore sperimentale
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei fattori sperimentali
Descrizione	1. L'amministratore chiede di modificare i dati di un fattore sperimentale 2. Il sistema raccoglie i dati archiviati riguardo al fattore sperimentale prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti del fattore sperimentale 4. L'amministratore apporta modifiche ai dati del fattore sperimentale 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate al fattore sperimentale 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati del fattore sperimentale
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un fattore sperimentale con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.45 Delete experimental factor

Nome	Delete experimental factor
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un fattore sperimentale
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei fattori sperimentali
Descrizione	1. L'amministratore chiede di eliminare un fattore sperimentale 2. Il sistema raccoglie i dati archiviati riguardo al fattore sperimentale prescelto 3. Il sistema mostra all'amministratore i dati correnti del fattore sperimentale e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare il fattore sperimentale 5. Il sistema elimina il fattore sperimentale 6. Il sistema notifica all'amministratore l'avvenuta eliminazione del fattore sperimentale
Alternative	1. L'amministratore rinuncia all'eliminazione del fattore sperimentale e il sistema ripresenta all'amministratore la pagina di visualizzazione dei fattori sperimentali 2. Il sistema rileva la presenza di uno o più valori associati al fattore sperimentale e chiede di eliminare quei valori prima di eliminare il fattore sperimentale
Postcondizioni	Nessuna

3.7.4.46 Experiment types administration

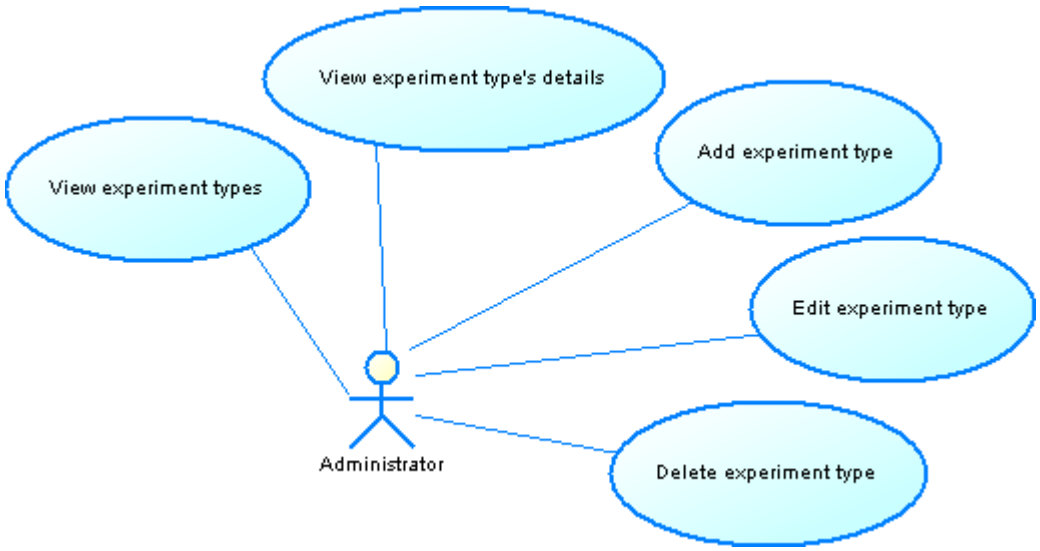

Figura 3.21.: Use case diagram "experiment type administration" dell'attore Administrator.

3.7.4.47 View experiment types

Nome	View experiment types
Iniziatore	Administrator
Scopo	Visualizzare le tipologie di esperimento archiviate
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare le tipologie di esperimento 2. Il sistema raccoglie i dati riguardanti tutte le tipologie di esperimento archiviate 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente le tipologie di esperimento
Alternative	1. Il sistema non trova tipologie di esperimento in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.48 View experiment type's details

Nome	View experiment type's details
Iniziatore	Administrator

Scopo	Visualizzare i dettagli relativi ad un tipo di esperimento
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle tipologie di esperimento
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un tipo di esperimento 2. Il sistema raccoglie i dati archiviati riguardo al tipo di esperimento prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sul tipo di esperimento
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.49 Add experiment type

Nome	Add experiment type
Iniziatore	Administrator
Scopo	Inserire un nuovo tipo di esperimento
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle tipologie di esperimento
Descrizione	1. L'amministratore chiede di inserire un nuovo tipo di esperimento 2. Il sistema mostra all'amministratore il form da compilare con i dati del tipo di esperimento 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo tipo di esperimento 5. Il sistema notifica all'amministratore l'avvenuto salvataggio del tipo di esperimento
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un tipo di esperimento con lo stesso nome e chiede che sia scelto un nome differente

Postcondizioni	Nessuna

3.7.4.50 Edit experiment type

Nome	Edit experiment type
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un tipo di esperimento
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle tipologie di esperimento
Descrizione	1. L'amministratore chiede di modificare i dati di un tipo di esperimento 2. Il sistema raccoglie i dati archiviati riguardo al tipo di esperimento prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti del tipo di esperimento 4. L'amministratore apporta modifiche ai dati del tipo di esperimento 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate al tipo di esperimento 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati del tipo di esperimento
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un tipo di esperimento con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.51 Delete experiment type

Nome	Delete experiment type
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un tipo di esperimento

Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle tipologie di esperimento
Descrizione	1. L'amministratore chiede di eliminare un tipo di esperimento 2. Il sistema raccoglie i dati archiviati riguardo al tipo di esperimento prescelto 3. Il sistema mostra all'amministratore i dati correnti del tipo di esperimento e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare il tipo di esperimento 5. Il sistema elimina il tipo di esperimento 6. Il sistema notifica all'amministratore l'avvenuta eliminazione del tipo di esperimento
Alternative	1. L'amministratore rinuncia all'eliminazione del tipo di esperimento e il sistema ripresenta all'amministratore la pagina di visualizzazione delle tipologie di esperimento 2. Il sistema rileva la presenza di uno o più esperimenti associati al tipo di esperimento e chiede di eliminare quegli esperimenti prima di eliminare il tipo di esperimento
Postcondizioni	Nessuna

3.7.4.52 Platform types administration

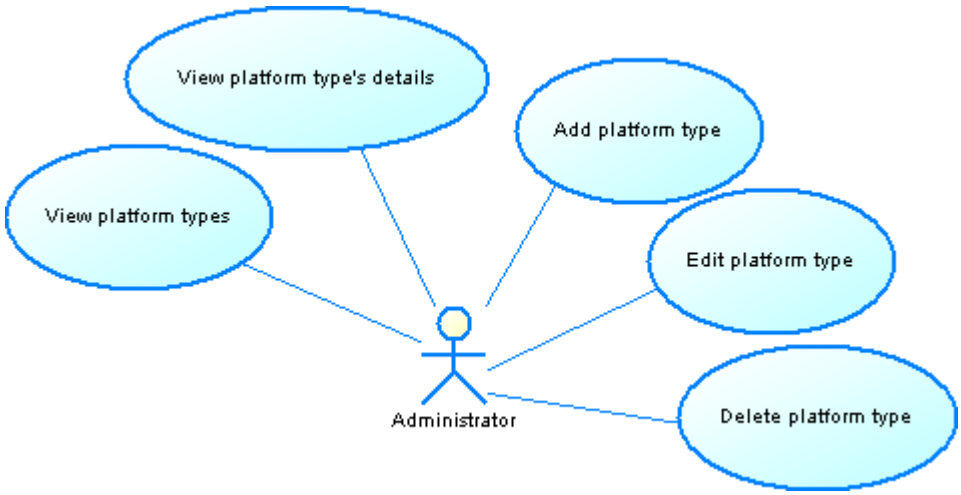

Figura 3.22.: Use case diagram "platform types administration" dell'attore Administrator.

3.7.4.53 View platform types

Nome	View platform types
Iniziatore	Administrator
Scopo	Visualizzare le tipologie di piattaforma archiviate
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare le tipologie di piattaforma 2. Il sistema raccoglie i dati riguardanti tutte le tipologie di piattaforma archiviate 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente le tipologie di piattaforma
Alternative	1. Il sistema non trova tipologie di piattaforma in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.54 View platform type's details

Nome	View platform type's details
Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad un tipo di piatta-

	forma
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle tipologie di piattaforma
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un tipo di piattaforma 2. Il sistema raccoglie i dati archiviati riguardo al tipo di piattaforma prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sul tipo di piattaforma
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.55 Add platform type

Nome	Add platform type
Iniziatore	Administrator
Scopo	Inserire un nuovo tipo di piattaforma
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle tipologie di piattaforma
Descrizione	1. L'amministratore chiede di inserire un nuovo tipo di piattaforma 2. Il sistema mostra all'amministratore il form da compilare con i dati del tipo di piattaforma 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo tipo di piattaforma 5. Il sistema notifica all'amministratore l'avvenuto salvataggio del tipo di piattaforma
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un tipo di piattaforma con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.56 Edit platform type

Nome	Edit platform type
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un tipo di piattaforma
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle tipologie di piattaforma
Descrizione	1. L'amministratore chiede di modificare i dati di un tipo di piattaforma 2. Il sistema raccoglie i dati archiviati riguardo al tipo di piattaforma prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti del tipo di piattaforma 4. L'amministratore apporta modifiche ai dati del tipo di piattaforma 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate al tipo di piattaforma 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati del tipo di piattaforma
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un tipo di piattaforma con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.57 Delete platform type

Nome	Delete platform type
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un tipo di piattaforma
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco delle tipologie di piattaforma

Descrizione	1. L'amministratore chiede di eliminare un tipo di piattaforma 2. Il sistema raccoglie i dati archiviati riguardo al tipo di piattaforma prescelto 3. Il sistema mostra all'amministratore i dati correnti del tipo di piattaforma e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare il tipo di piattaforma 5. Il sistema elimina il tipo di piattaforma 6. Il sistema notifica all'amministratore l'avvenuta eliminazione del tipo di piattaforma
Alternative	1. L'amministratore rinuncia all'eliminazione del tipo di piattaforma e il sistema ripresenta all'amministratore la pagina di visualizzazione delle tipologie di piattaforma 2. Il sistema rileva la presenza di una o più piattaforme associate al tipo di piattaforma e chiede di eliminare quelle piattaforme prima di eliminare il tipo di piattaforma
Postcondizioni	Nessuna

3.7.4.58 Organisms administration

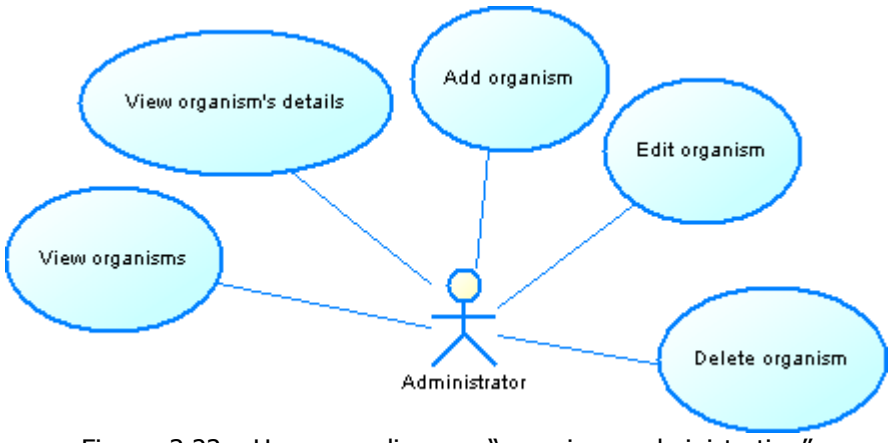

Figura 3.23.: Use case diagram "organisms administration" dell'attore Administrator.

3.7.4.59 View organisms

Nome	View organisms
Iniziatore	Administrator
Scopo	Visualizzare gli organismi archiviati
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare gli organismi 2. Il sistema raccoglie i dati riguardanti tutti gli organismi archiviati 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente gli organismi
Alternative	1. Il sistema non trova organismi in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.60 View organism's details

Nome	View organism's details
Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad un organismo
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco degli organismi

Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un organismo 2. Il sistema raccoglie i dati archiviati riguardo all'organismo prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sull'organismo
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.61 Add organism

Nome	Add organism
Iniziatore	Administrator
Scopo	Inserire un nuovo organismo
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco degli organismi
Descrizione	1. L'amministratore chiede di inserire un nuovo organismo 2. Il sistema mostra all'amministratore il form da compilare con i dati dell'organismo 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo organismo 5. Il sistema notifica all'amministratore l'avvenuto salvataggio dell'organismo
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un organismo con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.62 Edit organism

Nome	Edit organism
Iniziatore	Administrator

Scopo	Modificare i dati relativi ad un organismo
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco degli organismi
Descrizione	1. L'amministratore chiede di modificare i dati di un organismo 2. Il sistema raccoglie i dati archiviati riguardo all'organismo prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti dell'organismo 4. L'amministratore apporta modifiche ai dati dell'organismo 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate all'organismo 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati dell'organismo
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un organismo con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.63 Delete organism

Nome	Delete organism
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un organismo
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco degli organismi
Descrizione	1. L'amministratore chiede di eliminare un organismo 2. Il sistema raccoglie i dati archiviati riguardo all'organismo prescelto 3. Il sistema mostra all'amministratore i dati

	correnti dell'organismo e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare l'organismo 5. Il sistema elimina l'organismo 6. Il sistema notifica all'amministratore l'avvenuta eliminazione dell'organismo
Alternative	1. L'amministratore rinuncia all'eliminazione del organismo e il sistema ripresenta all'amministratore la pagina di visualizzazione degli organismi 2. Il sistema rileva la presenza di uno o più campioni associati all'organismo e chiede di eliminare quei campioni prima di eliminare l'organismo
Postcondizioni	Nessuna

3.7.4.64 Statistical tests administration

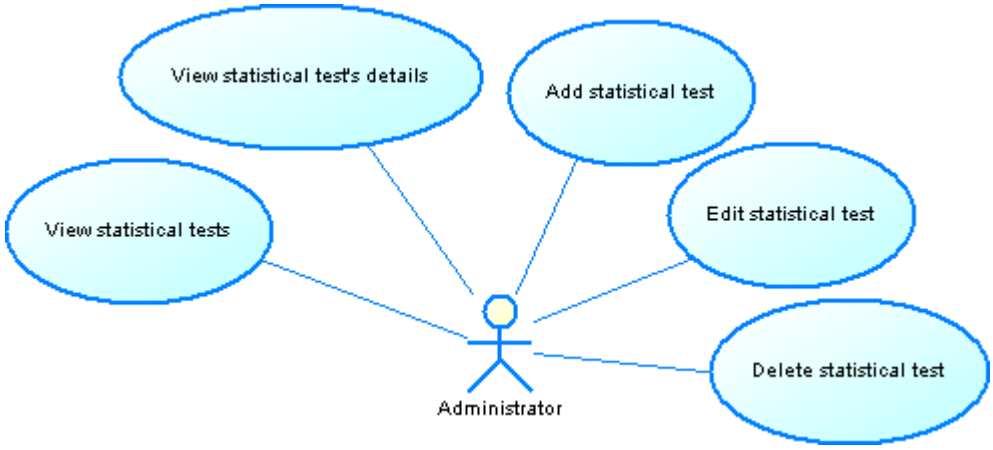

Figura 3.24.: Use case diagram "statistical test administration" dell'attore Administrator.

3.7.4.65 View statistical tests

Nome	View statistical tests
Iniziatore	Administrator
Scopo	Visualizzare i test statistici archiviati
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare i test statistici 2. Il sistema raccoglie i dati riguardanti tutti i test statistici archiviati 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente i test statistici
Alternative	1. Il sistema non trova test statistici in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.66 View statistical test's details

Nome	View statistical test's details
Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad un test statistico

Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei test statistici
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un test statistico 2. Il sistema raccoglie i dati archiviati riguardo al test statistico prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sul test statistico
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.67 Add statistical test

Nome	Add statistical test
Iniziatore	Administrator
Scopo	Inserire un nuovo test statistico
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei test statistici
Descrizione	1. L'amministratore chiede di inserire un nuovo test statistico 2. Il sistema mostra all'amministratore il form da compilare con i dati del test statistico 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo test statistico 5. Il sistema notifica all'amministratore l'avvenuto salvataggio del test statistico
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un test statistico con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.68 Edit statistical test

Nome	Edit statistical test
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un test statistico
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei test statistici
Descrizione	1. L'amministratore chiede di modificare i dati di un test statistico 2. Il sistema raccoglie i dati archiviati riguardo al test statistico prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti del test statistico 4. L'amministratore apporta modifiche ai dati del test statistico 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate al test statistico 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati del test statistico
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un test statistico con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.69 Delete statistical test

Nome	Delete statistical test
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un test statistico
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei test statistici
Descrizione	1. L'amministratore chiede di eliminare un test

	statistico 2. Il sistema raccoglie i dati archiviati riguardo al test statistico prescelto 3. Il sistema mostra all'amministratore i dati correnti del test statistico e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare il test statistico 5. Il sistema elimina il test statistico 6. Il sistema notifica all'amministratore l'avvenuta eliminazione del test statistico
Alternative	1. L'amministratore rinuncia all'eliminazione del test statistico e il sistema ripresenta all'amministratore la pagina di visualizzazione dei test statistici 2. Il sistema rileva la presenza di una o più analisi associate al test statistico e chiede di eliminare quelle analisi prima di eliminare il test statistico
Postcondizioni	Nessuna

3.7.4.70 Statistical parameters administration

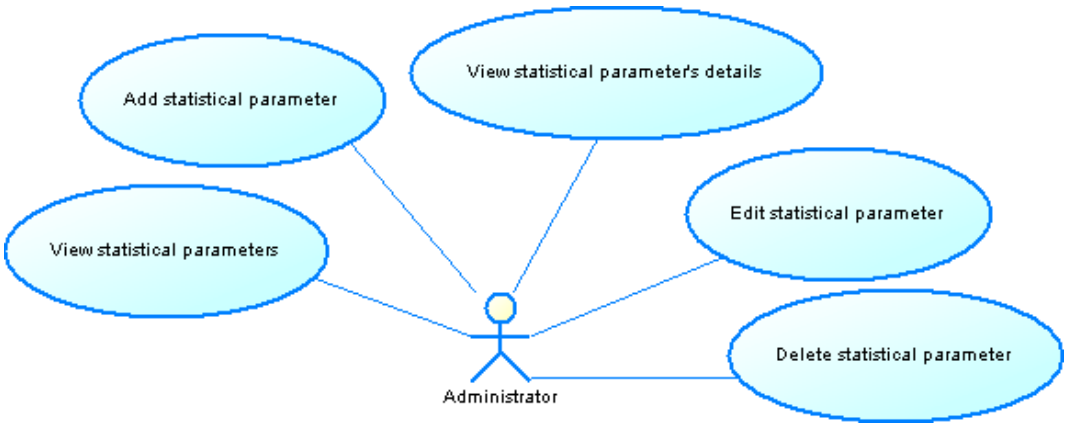

Figura 3.25.: Use case diagram "statistical parameters administration" dell'attore Administrator.

3.7.4.71 View statistical parameters

Nome	View statistical parameters
Iniziatore	Administrator
Scopo	Visualizzare i parametri statistici archiviati
Precondizioni	Nessuna
Descrizione	1. L'amministratore chiede di visualizzare i parametri statistici 2. Il sistema raccoglie i dati riguardanti tutti i parametri statistici archiviati 3. Il sistema mostra all'amministratore una tabella, eventualmente distribuita su più pagine, contenente i parametri statistici
Alternative	1. Il sistema non trova parametri statistici in archivio e mostra un messaggio di notifica dell'evento
Postcondizioni	Nessuna

3.7.4.72 View statistical parameter's details

Nome	View statistical parameter's details
Iniziatore	Administrator
Scopo	Visualizzare i dettagli relativi ad un parametro statistico

Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei parametri statistici
Descrizione	1. L'amministratore chiede di visualizzare i dettagli di un parametro statistico 2. Il sistema raccoglie i dati archiviati riguardanti il parametro statistico prescelto 3. Il sistema mostra all'amministratore tutte le informazioni disponibili sul parametro statistico
Alternative	Nessuna
Postcondizioni	Nessuna

3.7.4.73 Add statistical parameter

Nome	Add statistical parameter
Iniziatore	Administrator
Scopo	Inserire un nuovo parametro statistico
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei parametri statistici
Descrizione	1. L'amministratore chiede di inserire un nuovo parametro statistico 2. Il sistema mostra all'amministratore il form da compilare con i dati del parametro statistico 3. L'amministratore inserisce i nuovi dati 4. Il sistema salva il nuovo parametro statistico 5. Il sistema notifica all'amministratore l'avvenuto salvataggio del parametro statistico
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un parametro statistico con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.74 Edit statistical parameter

Nome	Edit statistical parameter
Iniziatore	Administrator
Scopo	Modificare i dati relativi ad un parametro statistico
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei parametri statistici
Descrizione	1. L'amministratore chiede di modificare i dati di un parametro statistico 2. Il sistema raccoglie i dati archiviati riguardo al parametro statistico prescelto 3. Il sistema mostra all'amministratore un form precompilato con i dati correnti del parametro statistico 4. L'amministratore apporta modifiche ai dati del parametro statistico 5. L'amministratore inserisce i nuovi dati 6. Il sistema salva le modifiche apportate al parametro statistico 7. Il sistema notifica all'amministratore l'avvenuta modifica dei dati del parametro statistico
Alternative	1. Il sistema si ravvede della presenza di campi obbligatori non compilati o di dati non conformi al formato prestabilito e ripresenta il form all'amministratore includendo messaggi esplicativi dei problemi incontrati 2. Il sistema rileva la presenza in archivio di un parametro statistico con lo stesso nome e chiede che sia scelto un nome differente
Postcondizioni	Nessuna

3.7.4.75 Delete statistical parameter

Nome	Delete statistical parameter
Iniziatore	Administrator
Scopo	Eliminare i dati relativi ad un parametro statistico
Precondizioni	Il sistema ha mostrato all'amministratore l'elenco dei parametri statistici

Descrizione	1. L'amministratore chiede di eliminare un parametro statistico 2. Il sistema raccoglie i dati archiviati riguardo al parametro statistico prescelto 3. Il sistema mostra all'amministratore i dati correnti del parametro statistico e chiede conferma dell'eliminazione 4. L'amministratore sceglie di eliminare il parametro statistico 5. Il sistema elimina il parametro statistico 6. Il sistema notifica all'amministratore l'avvenuta eliminazione del parametro statistico
Alternative	1. L'amministratore rinuncia all'eliminazione del parametro statistico e il sistema ripresenta all'amministratore la pagina di visualizzazione dei parametri statistici 1. Il sistema rileva la presenza di uno o più valori associati al parametro statistico e chiede di eliminare quei valori prima di eliminare il parametro statistico
Postcondizioni	Nessuna

3.7.4.76 Experiments administration

L'amministratore eredita tutti i casi d'uso già presentati per l'attore User per quanto attiene l'amministrazione degli esperimenti e di campioni e analisi ad essi relativi. Fanno eccezione, ovviamente:

- tutte le affermazioni relative all'ambito di visibilità dell'utente che, nel caso di un amministratore, è esteso a tutti gli esperimenti archiviati;
- tutte le affermazioni relative alla proprietà dell'esperimento in quanto gli amministratori hanno accesso in ogni caso alla mofica, eliminazione e indicizzazione di ogni dato archiviato.

3.7.4.77 Index administration

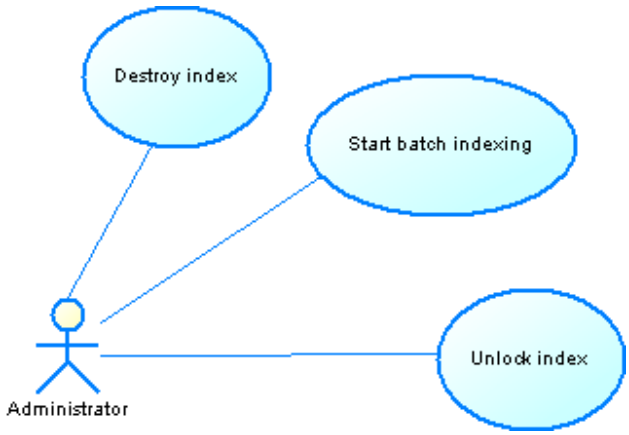

Figura 3.26.: Use case diagram "index administration" dell'attore Administrator.

3.7.4.78 Destroy index

Nome	Destroy index
Iniziatore	Administrator
Scopo	Distruggere l'indice dei dati e crearne uno nuovo
Precondizioni	L'indice risulta danneggiato o non più allineato ai dati archiviati dal database
Descrizione	1. L'amministratore chiede di distruggere l'indice 2. Il sistema distrugge l'indice 3. Il sistema crea un nuovo indice vuoto

	4. Il sistema notifica all'amministratore l'avvenuta distruzione dell'indice
Alternative	Nessuna
Postcondizioni	Un nuovo indice vuoto è stato creato

3.7.4.79 Start batch indexing

Nome	Start batch indexing
Iniziatore	Administrator
Scopo	Lanciare manualmente il processo di indicizzazione dei dati non ancora indicizzati
Precondizioni	L'amministratore ha bisogno che tutti i dati archiviati siano disponibili per la ricerca
Descrizione	1. L'amministratore chiede di indicizzare subito tutti i dati non ancora indicizzati 2. Il sistema indicizza uno per volta tutti i dati non indicizzati 3. Il sistema crea un log del processo di indicizzazione e lo invia con una e-mail all'amministratore 4. Il sistema notifica all'amministratore il completamento del processo di indicizzazione
Alternative	1. Alcuni dati non sono indicizzabili perché non rispondenti al formato accettato dall'applicazione. 2. Il sistema invia all'utente proprietario dei dati non indicizzabili una e-mail di avviso
Postcondizioni	Tutti i dati archiviati nel sistema sono indicizzati e disponibili per la ricerca

3.7.4.80 Unlock index

Nome	Unlock index
Iniziatore	Administrator
Scopo	Eliminare un write lock pendente sull'indice
Precondizioni	L'indice risulta bloccato perché, a causa di un grave problema hardware, un processo di indicizzazione non è stato completato

Descrizione	1. L'amministratore chiede di sbloccare l'indice 2. Il sistema effettua un unlock forzato dell'indice 3. Il sistema notifica all'amministratore l'avvenuto unlock dell'indice
Alternative	Nessuna
Postcondizioni	L'indice è ora utilizzabile per la ricerca e l'indicizzazione

3.8 Data Transfer Objects, DTO

L'utilizzo simultaneo di una piattaforma object oriented, quale è JavaEE v5.0, e di un RDBMS ci impone un ulteriore passo progettuale consistente nel mapping delle entità individuate nel modello ER verso oggetti, cioè classi Java. Abbiamo bisogno di oggetti che si possano riempire a partire dalle tuple presenti nelle tabelle del data base, e che, viceversa, possano essere utilizzati agevolmente per riempire le tuple stesse. Tali oggetti devono essere dotati di attributi di tipo compatibile con i Data Types scelti nella costruzione del modello fisico dei dati.

Ci viene in aiuto, in questo senso, il pattern progettuale detto Data Transfer Objects (DTO), o anche Value Objects (VO). Il pattern DTO consiste nella creazione di oggetti, i DTO appunto, al solo scopo di trasferire dati tra diversi sottosistemi di una stessa applicazione. I DTO sono spesso utilizzati in congiunzione con i Business Objects, quegli oggetti che modellano il comportamento degli oggetti del mondo reale occupandosi anche dell'interazione col database.

La differenza tra i Data Transfer Objects e i Business Objects è che un DTO non è dotato di un comportamento eccetto quanto è necessario all'archiviazione e recupero dei suoi stessi dati, cioè utilizzando la terminologia dei JavaBeans, un DTO è dotato solo di attributi privati e metodi pubblici getters e setters.

Abbiamo creato un DTO per ogni tabella memorizzata nel nostro database. Ogni DTO avrà un insieme di attributi che mappano uno ad uno le colonne della tabella cui si riferisce, e un ulteriore insieme di attributi di supporto alle operazioni di join tra tabelle. Il DTO relativo ad una tabella contenente una chiave straniera conterrà anche tutti gli attributi utili a raccogliere informazioni dalla tabella cui la chiave straniera fa riferimento.

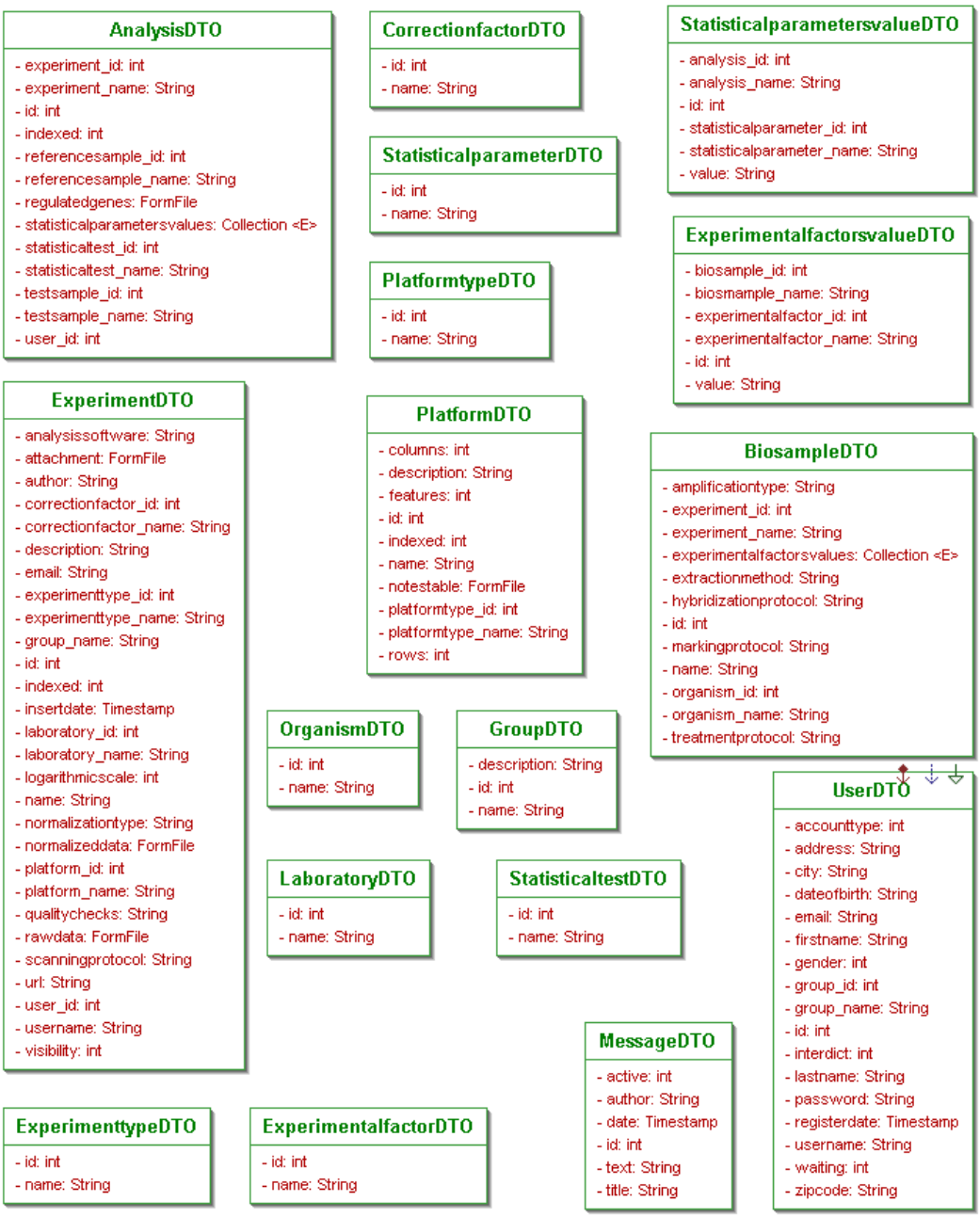

Figura 3.27.: I Data Transfer Objects creati per facilitare il trasferimento dei dati da e verso il Data Base.

La scelta dei nomi degli attributi, comunque, dovrebbe essere in grado di esplicare meglio il concetto. Nella Figura 3.27 è possibile osservare tutti i DTO creati per l'utilizzo nel sistema. L'assenza di legami tra gli oggetti presentati è dovuto essenzialmente al futuro utilizzo di questi oggetti: semplici contenitori da utilizzare per il

trasporto di dati da uno strato all'altro del sistema. Per motivi di spazio sono stati nascosti alla vista i metodi pubblici getter e setter indispensabili per l'accesso ad ogni attributo privato.

3.9 Mapping dei casi d'uso su Struts

Dalla lettura del paragrafo sui casi d'uso dovrebbe essere risultato evidente al lettore come ad ogni tabella archiviata nel database siano state associate per lo più operazioni di creazione, lettura, modifica ed eliminazione dei dati in essa contenuti. Tali operazioni, normalmente dette di CRUD (Create Retrieve Update e Delete) costituiscono il nocciolo di ogni applicazione data centrica. Ciò non poteva essere ignorato dai progettisti di un buon framework, infatti, una volta individuati i casi d'uso, è possibile passare direttamente al mapping di questi su Struts, una operazione che, come vedremo, può essere tranquillamente standardizzata.

3.9.1 I componenti base di Struts

Apache Struts non solo ci fornisce il controller da utilizzare per realizzare il pattern MVC nelle nostre applicazioni, ma ci fornisce anche un insieme di classi e interfacce a partire dalle quali sarà costruito lo scheletro della nostra applicazione web. Le classi che utilizzeremo frequentemente sono:

- org.apache.struts.action.Action che sarà ereditata da tutte le nostre Action, le classi alle quali la ActionServlet delega l'elaborazione delle richieste (request);
- org.apache.struts.action.ActionForm che sarà ereditata da tutti i nostri FormBean, oggetti contenitori che vengono popolati automaticamente dal framework a partire dai form contenuti nelle request;
- org.apache.struts.action.ActionMapping che contiene gli oggetti associati ad ogni Action nello struts-config.xml, in particolare gli ActionForward;

- org.apache.struts.action.ActionForward che contengono i path ai quali la servlet di Struts inoltra il flusso in base alla logica dell'applicazione.

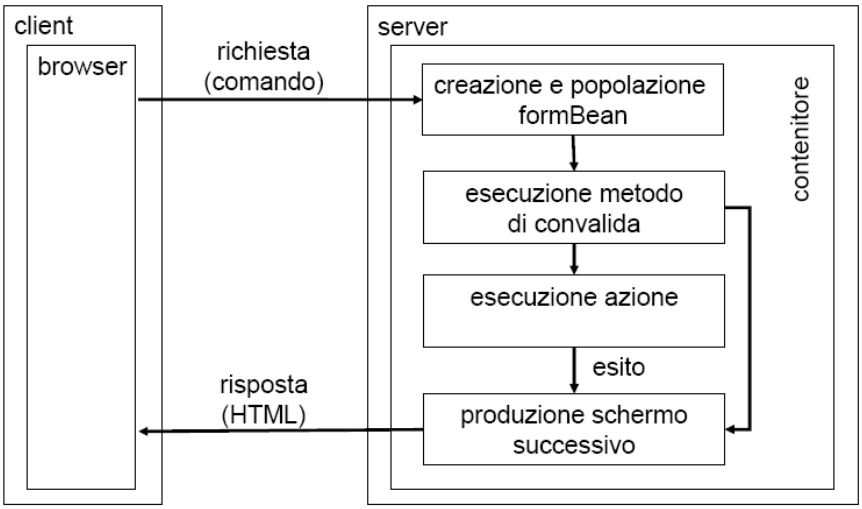

Figura 3.28.: Ciclo di vita di una request HTTP in Struts.

Brevemente possiamo dire che ogni volta che vorrà compiere una operazione l'utente accederà ad una pagina JSP, compilerà un form, ne effettuerà il submit, producendo di fatto una request HTTP che sarà intercettata dal controller di Struts, la ActionServlet. Il controller individuerà quale Action dovrà gestire la request a partire da quanto codificato nel file di configurazione di Struts.

La Action prescelta gestirà la request prelevando i parametri immagazzinati nel FormBean ad essa associato ed effettuerà una operazione di business su di essi. Al termine della sua esecuzione la Action restituirà al controller un oggetto, detto ActionForward, col quale sceglierà verso quale pagina o ulteriore Action dovrà essere indirizzata l'elaborazione. Al termine delle operazioni innescate dalla request dell'utente, il controller sceglierà l'opportuna pagina JSP che sarà tradotta in una pagina HTML inviata al browser dell'utente in una response HTTP.

3.9.2 Il file di configurazione di Struts

In generale, quindi, per mappare la maggior parte dei nostri use case su Apache Struts, avremo bisogno dei seguenti oggetti:

- una classe Action,
- un FormBean,
- uno o più ActionForward,
- una o più pagine JSP.

L'utilizzo di questi oggetti viene dichiarato nel file di configurazione di Struts, di solito struts-config.xml, un file XML di cui Exadel Studio Pro da una utilissima interpretazione grafica.

Bisogna notare, comunque, come tale impostazione sia stata applicata con alcune varianti:

- Molte Action class, coinvolte in operazioni di CRUD, estendono org.apache.struts.actions.DispatchAction in luogo della più comune org.apache.struts.action.Action. L'utilizzo di DispatchAction consente di raggruppare operazioni distinte, effettuate su uno stesso FormBean, in un unica Action class, interferendo sul mapping uno ad uno tra use case e Action;
- Molti FormBean estendono org.apache.struts.validator.ValidatorForm in luogo della più comune org.apache.struts.action.ActionForm. ValidatorForm realizza l'integrazione di Validator in struts automatizzando la validazione dei form e l'invio dei messaggi di notifica. Per questo nei grafici a seguire non c'è traccia del processo di validazione;
- Alcune Action generano un ActionForward indirizzato a forward di tipo globale, cioè comune a tutte le Action.

3.9.3 Le Actions create

3.9.3.1 Il simbolismo utilizzato

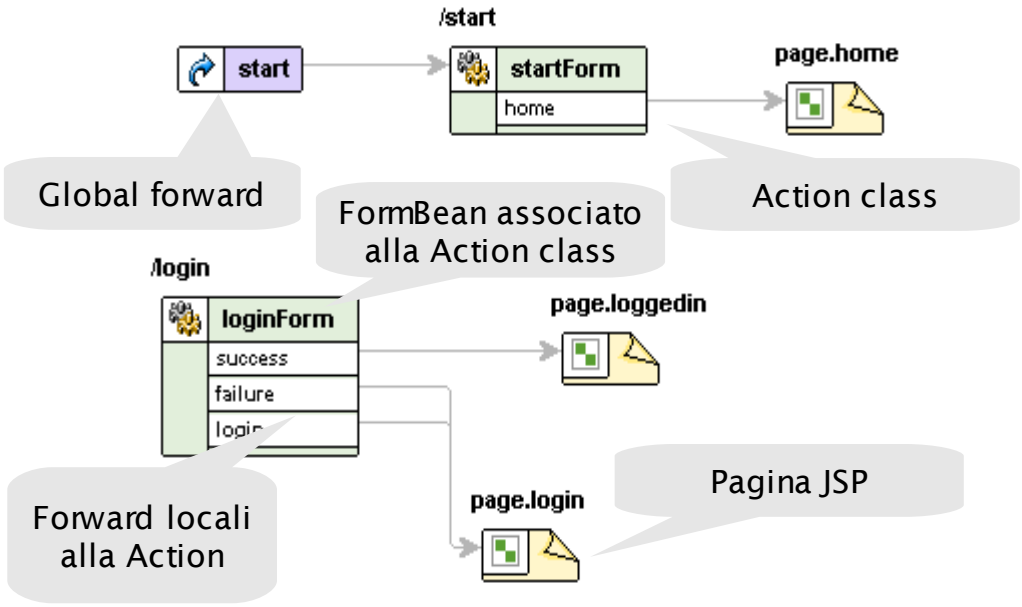

Figura 3.29.: Leggenda del formalismo utilizzato da Exadel Studio Pro nella rappresentazione del file di configurazione di Struts.

In figura 3.9 è riportata una leggenda che chiarisce il significato associato ai simboli utilizzati da Exadel Studio Pro del trarre un diagramma dal file di configurazione di Struts.A seguire sono riportate le rappresentazioni grafiche di alcune delle Action create in risposta ai casi d'uso definiti nei paragrafi precedenti.

3.9.3.2 Join

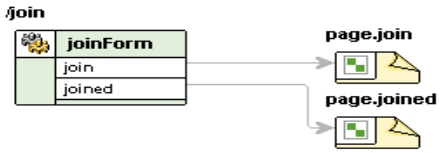

Figura 3.30.: Action mapping per le action join.

3.9.3.3 Log in

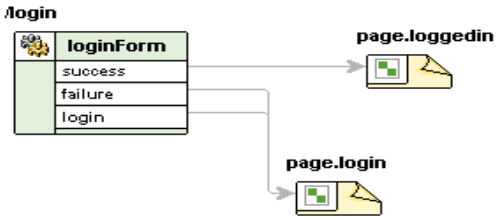

Figura 3.31.: Action mapping per la action login.

3.9.3.4 Password recovery

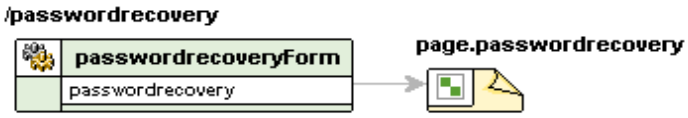

Figura 3.32.: Action mapping per la action passwordrecovery.

3.9.3.5 View home

Figura 3.33.: Action mapping per la action start.

3.9.3.6 Add experiment

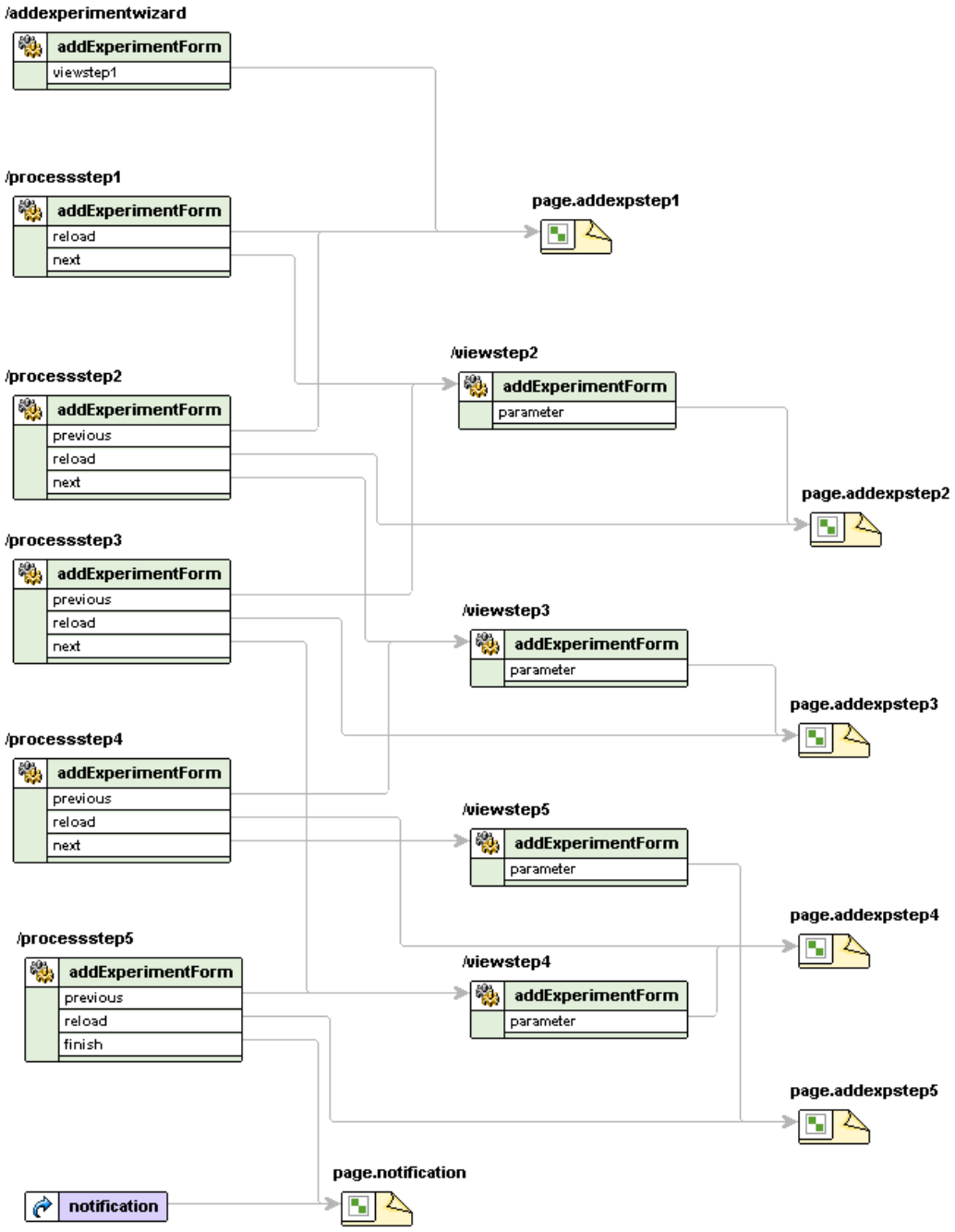

Figura 3.34.: Action mapping per le action che operano attorno al wizard per l'inserimento di un esperimento.

3.9.3.7 Log out

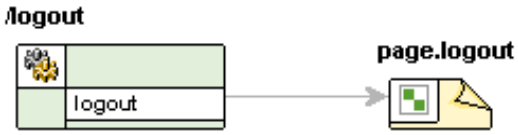

Figura 3.35.: Action mapping per la action logout.

3.9.3.8 View/Edit user account

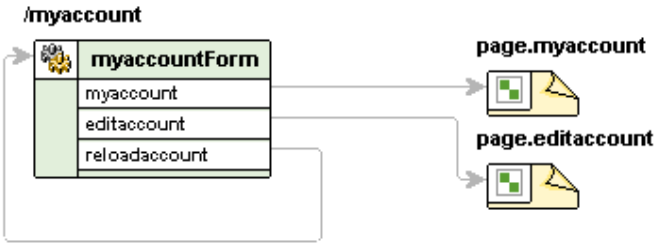

Figura 3.36.: Action mapping per la action myaccount.

3.9.3.9 Search

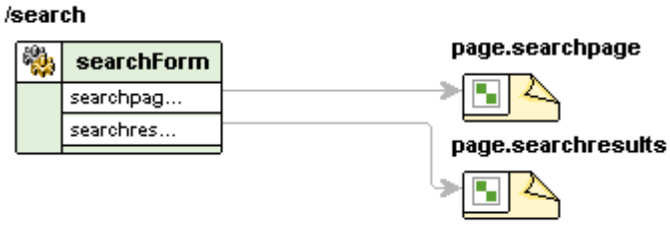

Figura 3.37.: Action mapping per la action search.

3.9.3.10 View platforms

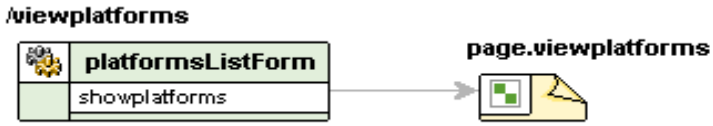

Figura 3.38.: Action mapping per la action viewplatforms.

3.9.3.11 Experiments administration

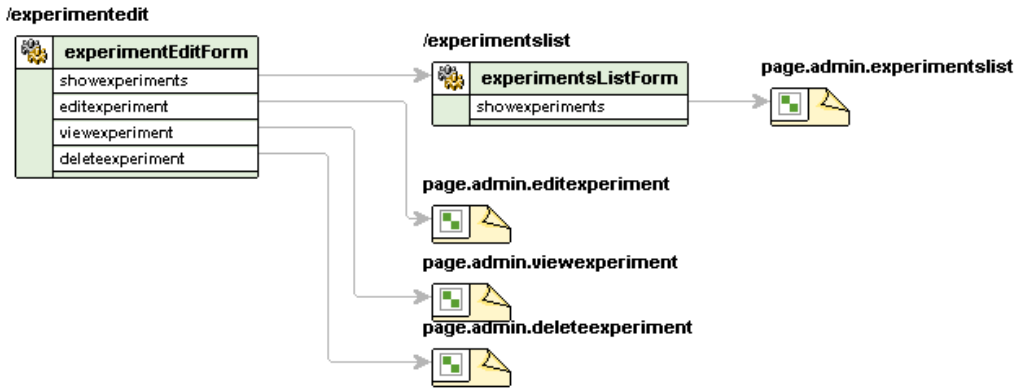

Figura 3.39.: Action mapping per le action experimentedit e experimentslist.

3.9.3.12 Analyses administration

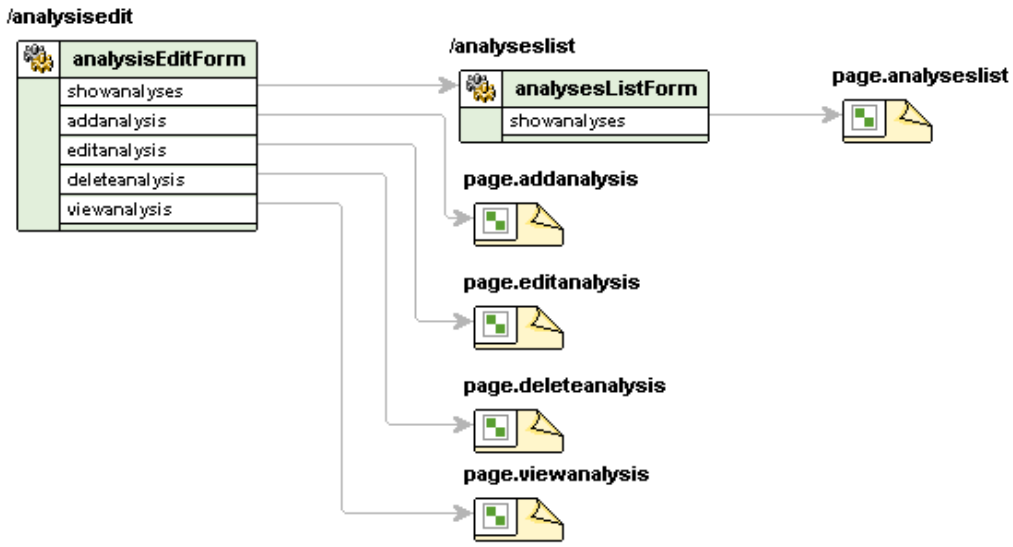

Figura 3.40.: Action mapping per le action analysisedit e analyseslist.

3.9.3.13 Biosamples administration

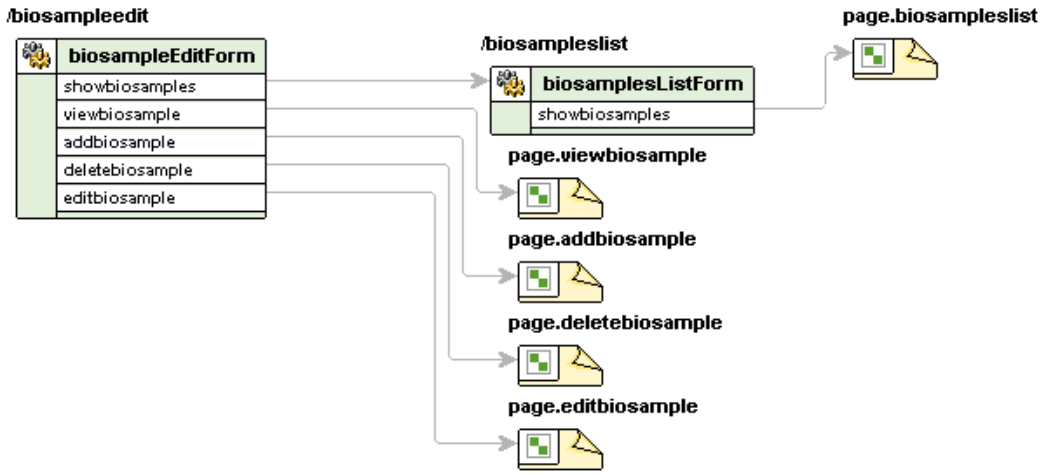

Figura 3.41.: Action mapping per le action biosampleedit e biosampleslist.

3.9.3.14 View with VIMIDA

Figura 3.42.: Action mapping per la action vimidaview.

3.9.3.15 Groups administration

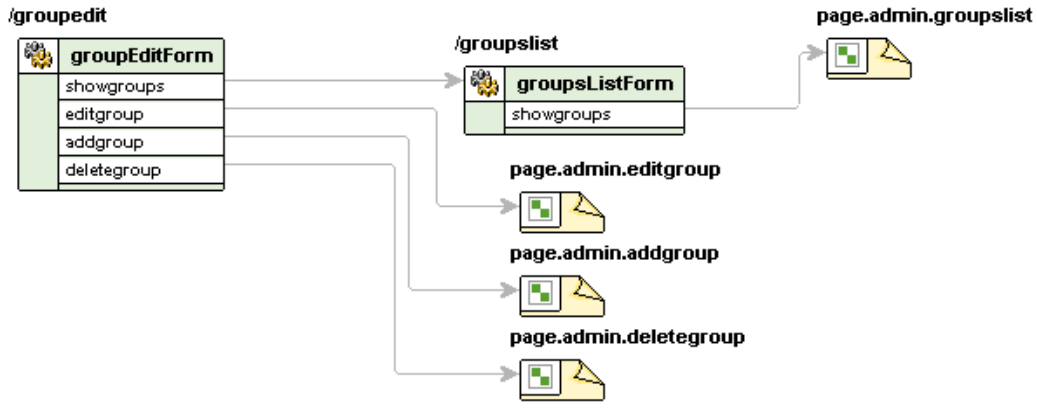

Figura 3.43.: Action mapping per le action groupedit e groupslist.

3.9.3.16 Users administration

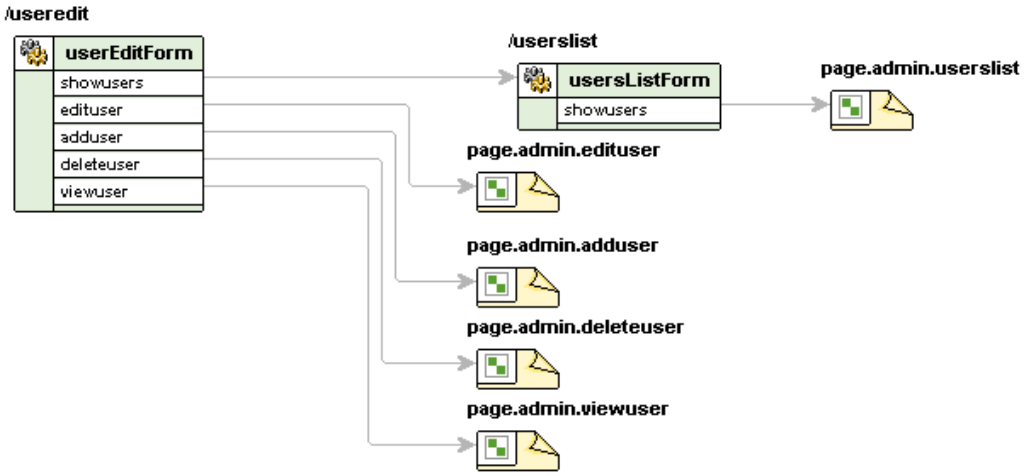

Figura 3.44.: Action mapping per le action useredit e userslist.

3.9.3.17 Platforms administration

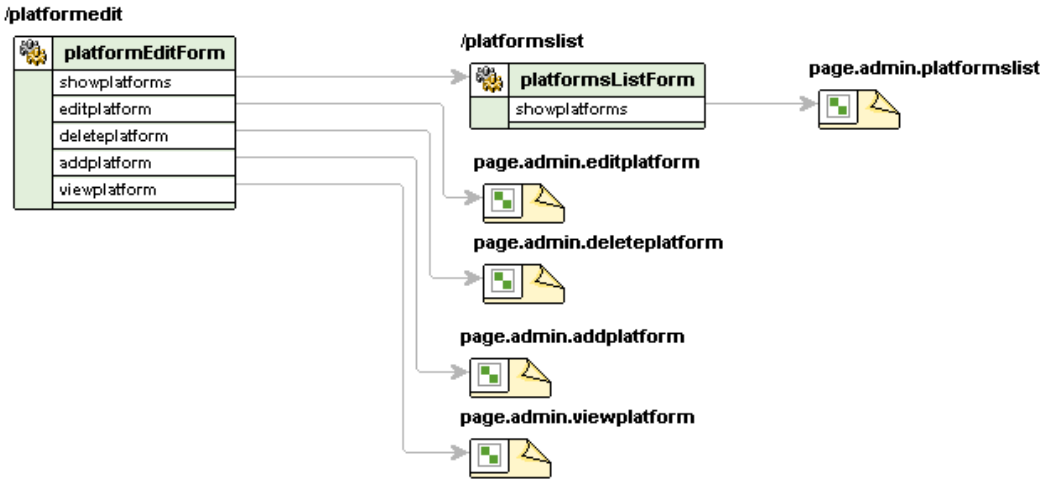

Figura 3.45.: Action mapping per le action platformedit e platformslist.

3.9.3.18 Messages administration

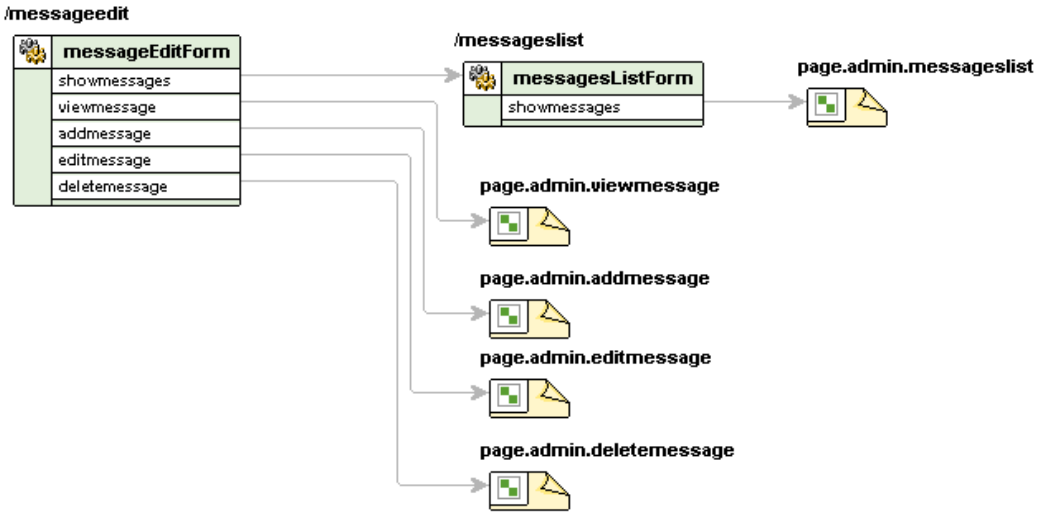

Figura 3.46.: Action mapping per le action messageedit e messageslist.

3.9.3.19 Laboratories administration

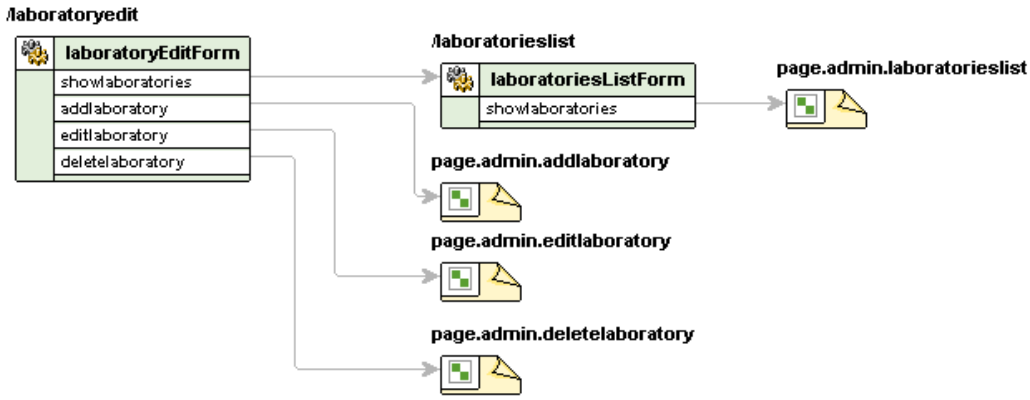

Figura 3.47.: Action mapping per le action laboratoryedit e laboratorieslist.

3.9.3.20 Correction factors administration

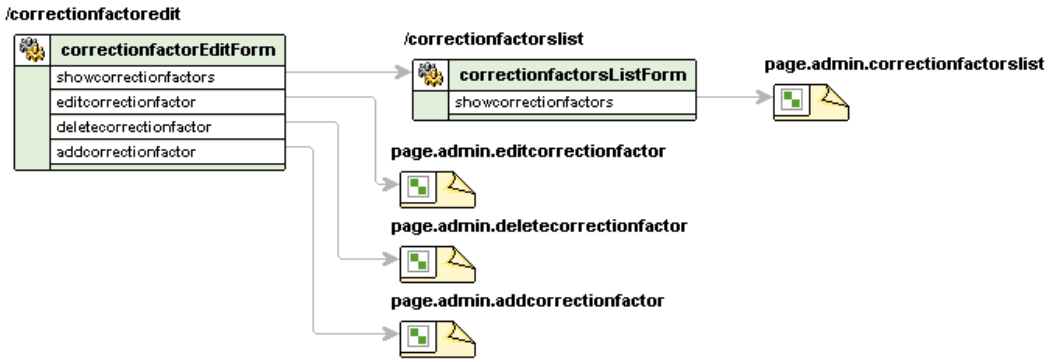

Figura 3.48.: Action mapping per le action correctionfactoredit e correctionfactorslist.

3.9.3.21 Experimental factors administration

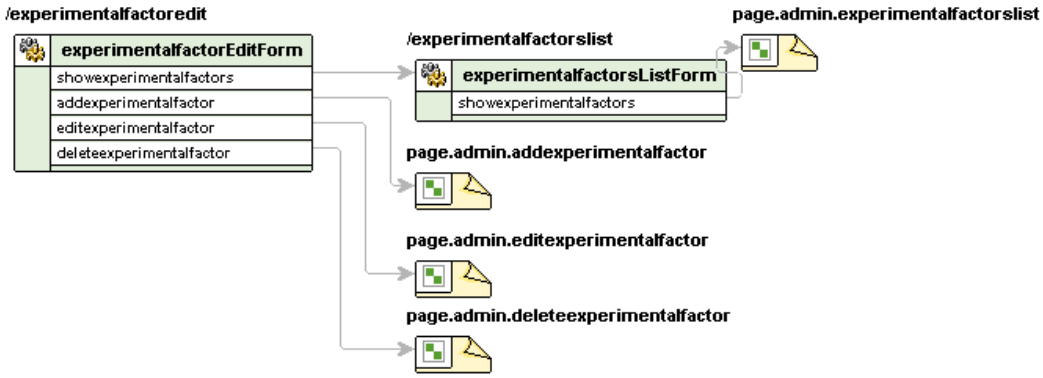

Figura 3.49.: Action mapping per le action experimentalfactoredit e experimentalfactorslist.

3.9.3.22 Experiment types administration

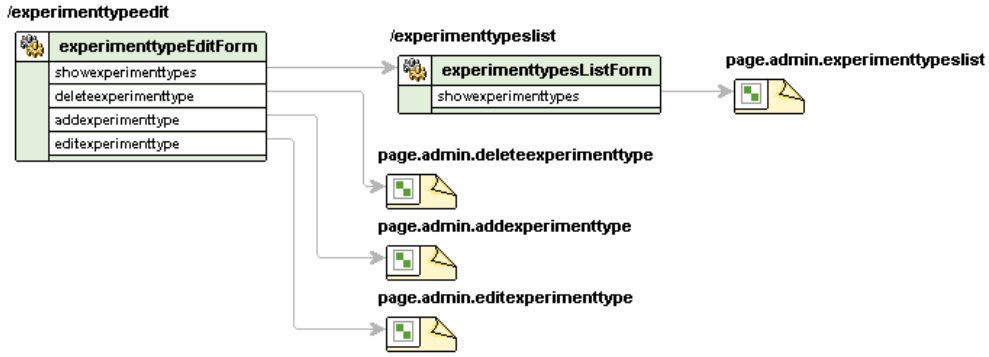

Figura 3.50.: Action mapping per le action experimenttypeedit e experimenttypeslist.

3.9.3.23 Platform types administration

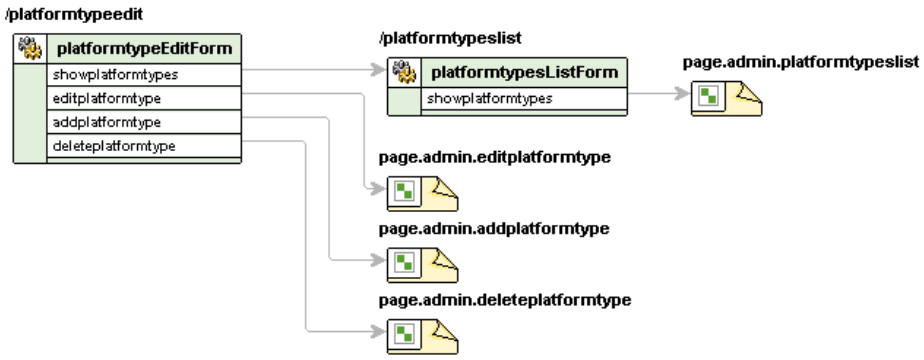

Figura 3.51.: Action mapping per le action platformtypeedit e platformtype-slist.

3.9.3.24 Organisms administration

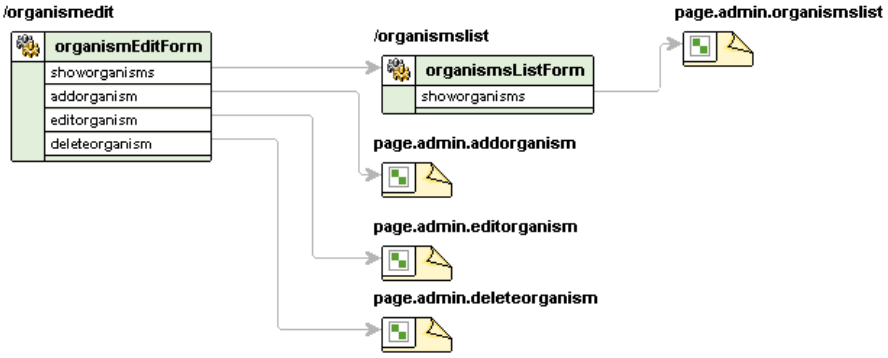

Figura 3.52.: Action mapping per le action oragnismedit e organismslist.

3.9.3.25 Statistical tests administration

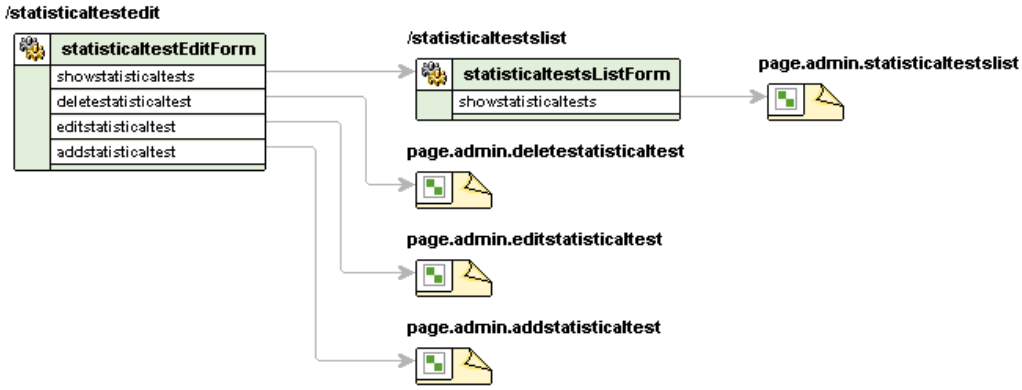

Figura 3.53.: Action mapping per le action statisticaltestedit e statisticaltestslist.

3.9.3.26 Statistical parameters administration

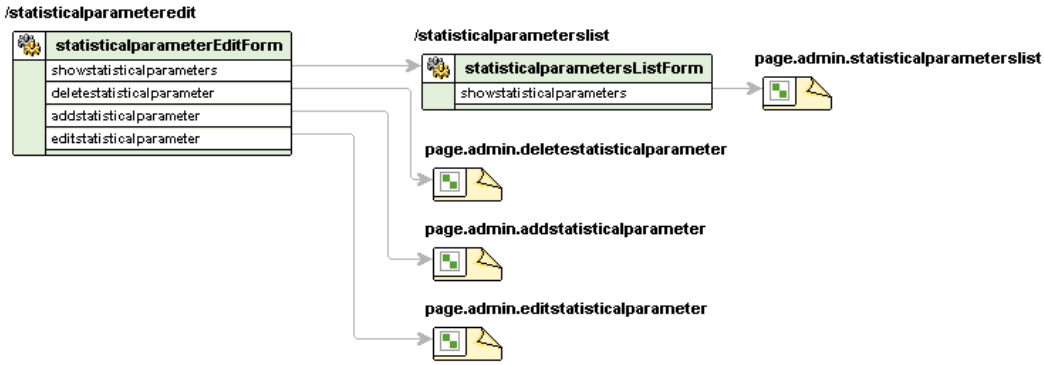

Figura 3.54.: Action mapping per le action statisticalparameteredit e statisticalparameterslist.

3.9.3.27 Index administration

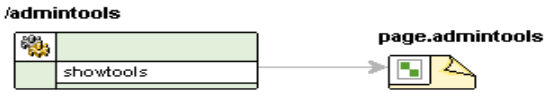

Figura 3.55.: Action mapping per la action admintools.

3.10 Class diagrams

Nelle illustrazioni incluse in questo paragrafo mostreremo alcuni class diagrams, in formato UML 2.0, ottenuti con eUML2, una plugin per Eclipse prodotta da Soyatec. Anche in questo caso sarà evidente l'effetto standardizzante di Struts sul design del sistema. Per motivi di semplicità e chiarezza, non tutte le relazioni esistenti tra le classi sono state esposte. Sono state privilegiate, in particolare, le interazioni tra Actions, FormBeans, DTO e BusinessObjects, tipiche dell'architettura di una applicazione web realizzata con Struts.

In molti diagrammi è evidente l'utilizzo della classe DispatchAction in luogo della classe Action. Nel funzionamento standard di Struts , il framework esegue il metodo execute() della Action che corrisponde al URL della richiesta effettuata dal client. Il metodo execute() costituisce quindi il punto di ingresso nel framework a fronte di una richiesta dal client. Questo funzionamento si adatta poco alle situazioni nelle quali è necessario eseguire una serie di elaborazioni tra loro logicamente collegate. Tipico è l'esempio di operazioni di inserimento, cancellazione , lettura e aggiornamento su una stessa tabella di un database.

Per evitare un inutile proliferare di classi action, Struts ci viene incontro fornendo DispatchAction. La DispatchAction è assolutamente analoga ad una Action base ma fornisce la possibilità di invocare diversi metodi della stessa purché il client specifichi il metodo da chiamare. In pratica è come una Action che non ha un solo metodo execute() ma ne ha n con nomi diversi. Il discorso è simile per la action LookupDispatchAction.

Per quanto riguarda i FormBean, invece, notiamo come ogni qual volta si sia voluto utilizzare il framework per la validazione Validator, sia stata estesa la classe ValidatorForm in luogo di ActionForm.

Infatti, per utilizzare Validator i FormBean dell'applicazione non devono estendere la classe ActionForm standard di Struts ma la classe ValidatorForm che fornisce una propria implementazione del metodo validate(). In questo caso non è più necessario scrivere il codice di validazione perché è Validator che lo fa per noi.

3.10.1.1 Join

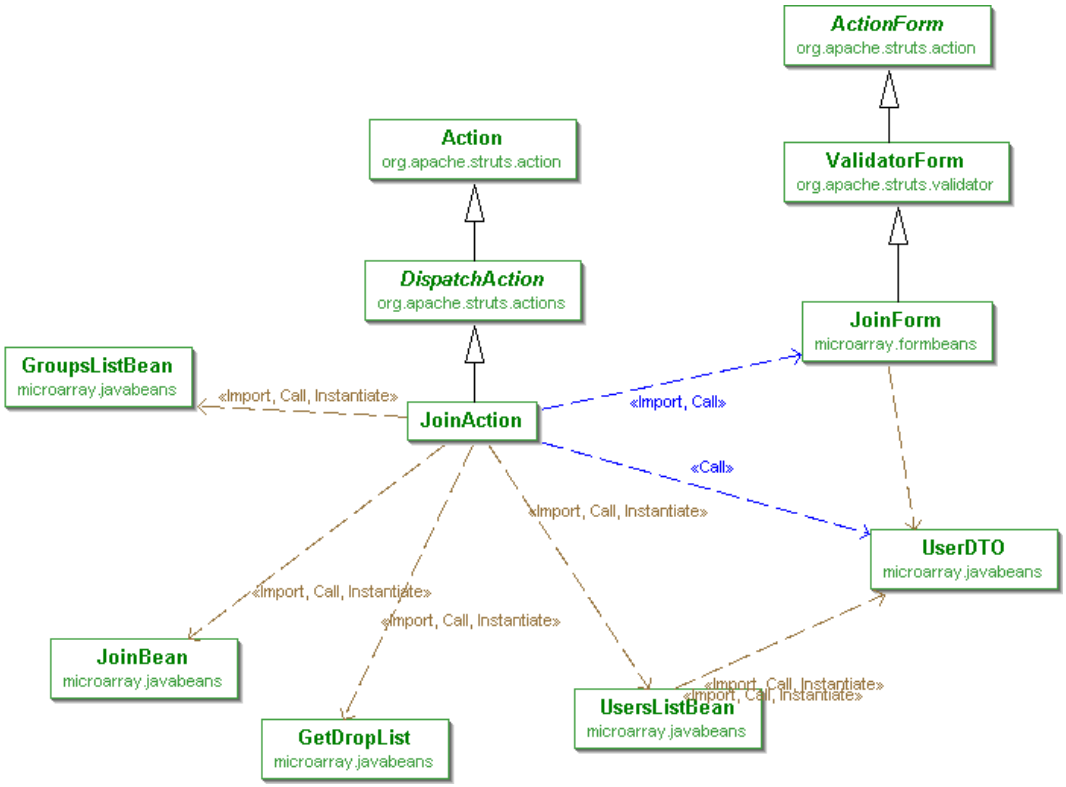

Figura 3.56.: Class diagram costruito attorno a JoinAction.

3.10.1.2 Log in

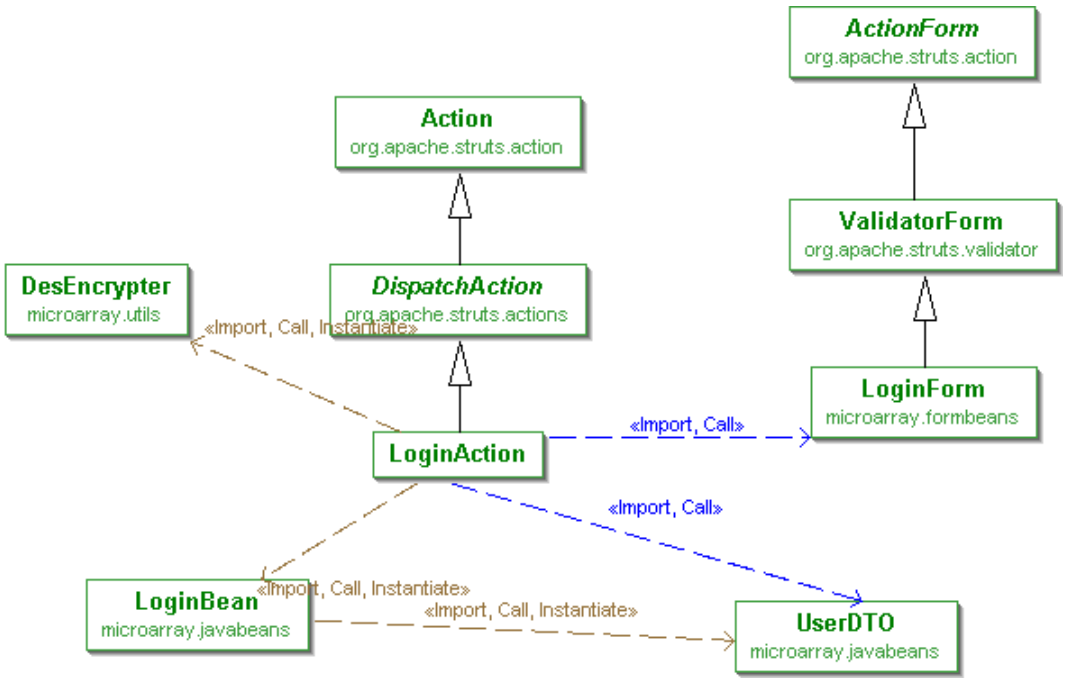

Figura 3.57.: Class diagram costruito attorno a LoginAction.

3.10.1.3 Password recovery

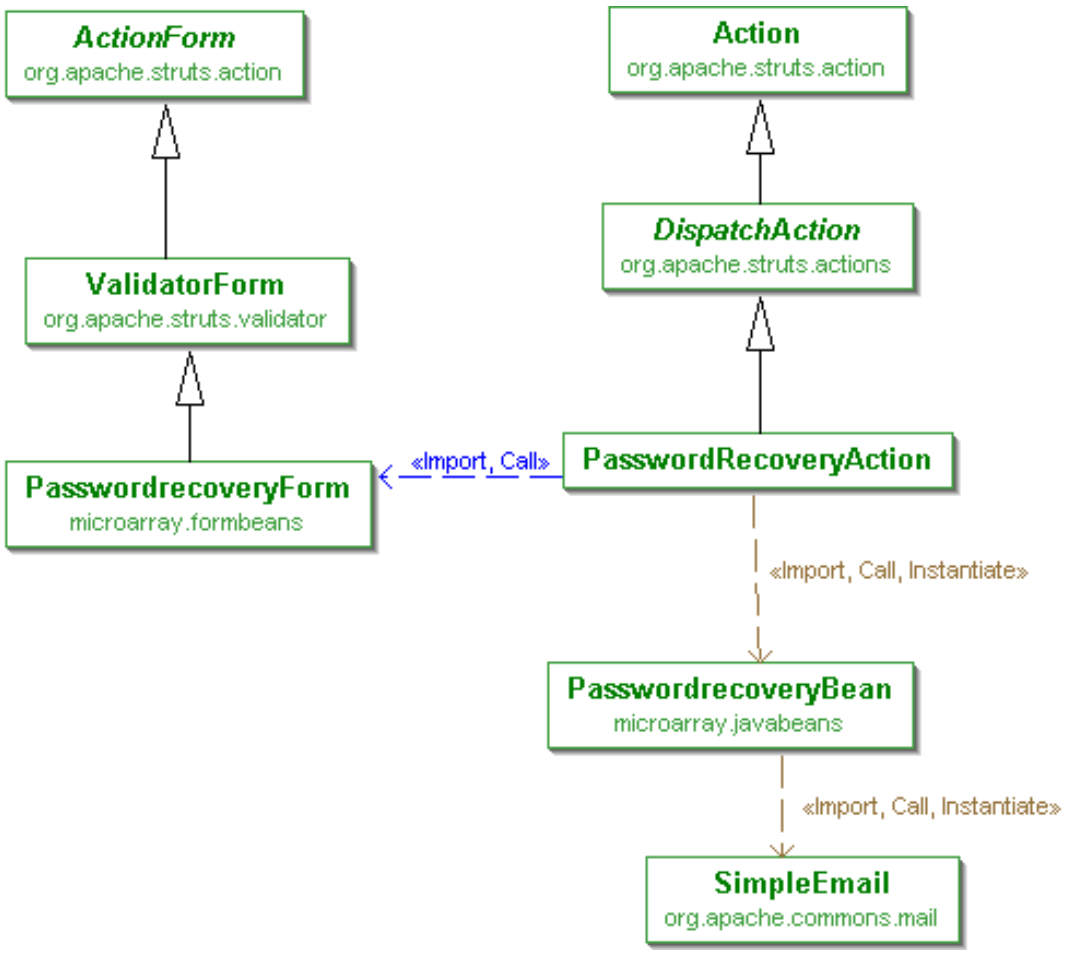

Figura 3.58.: Class diagram costruito attorno a PasswordRecoveryAction.

3.10.1.4 Add experiment

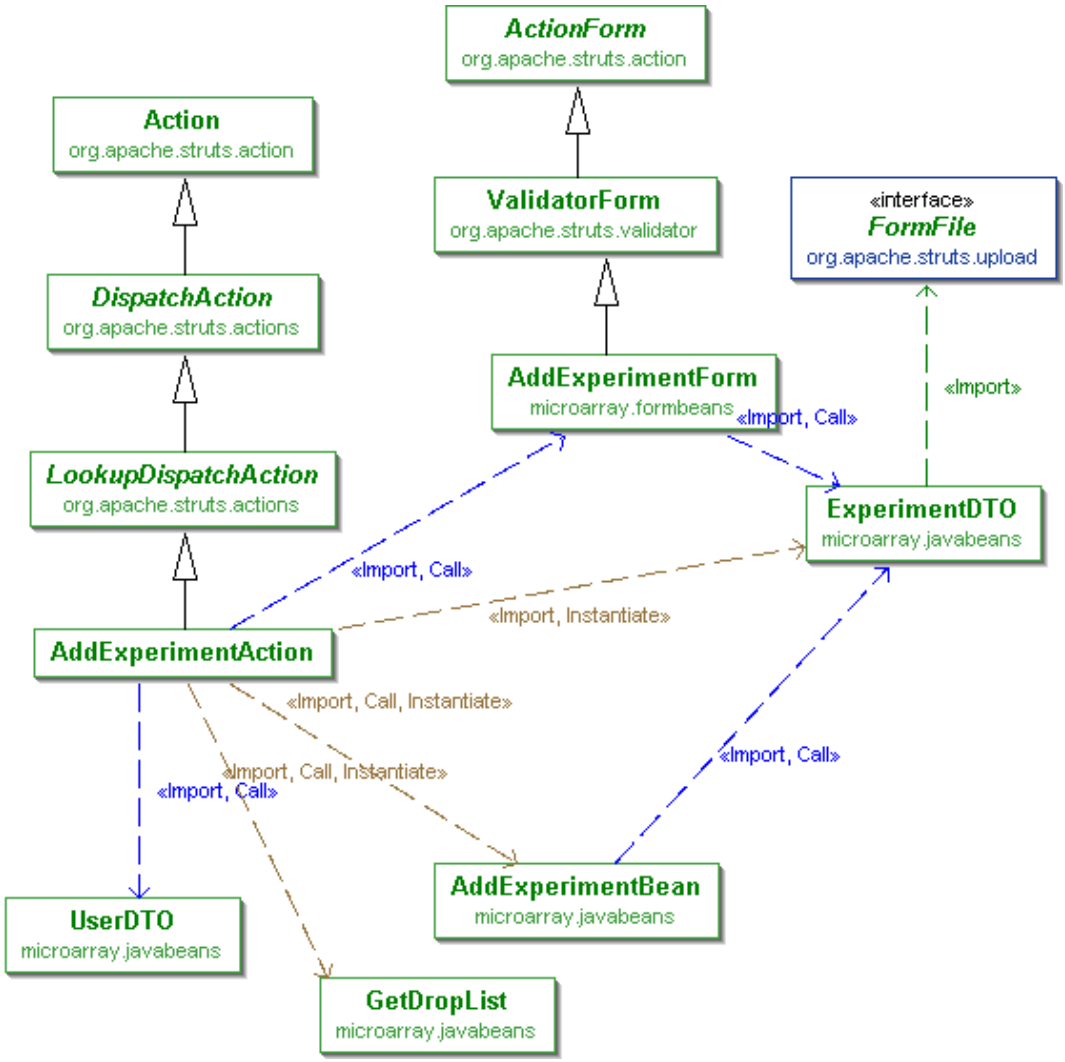

Figura 3.59.: Class diagram costruito attorno a AddExperimentAction.

3.10.1.5 Search

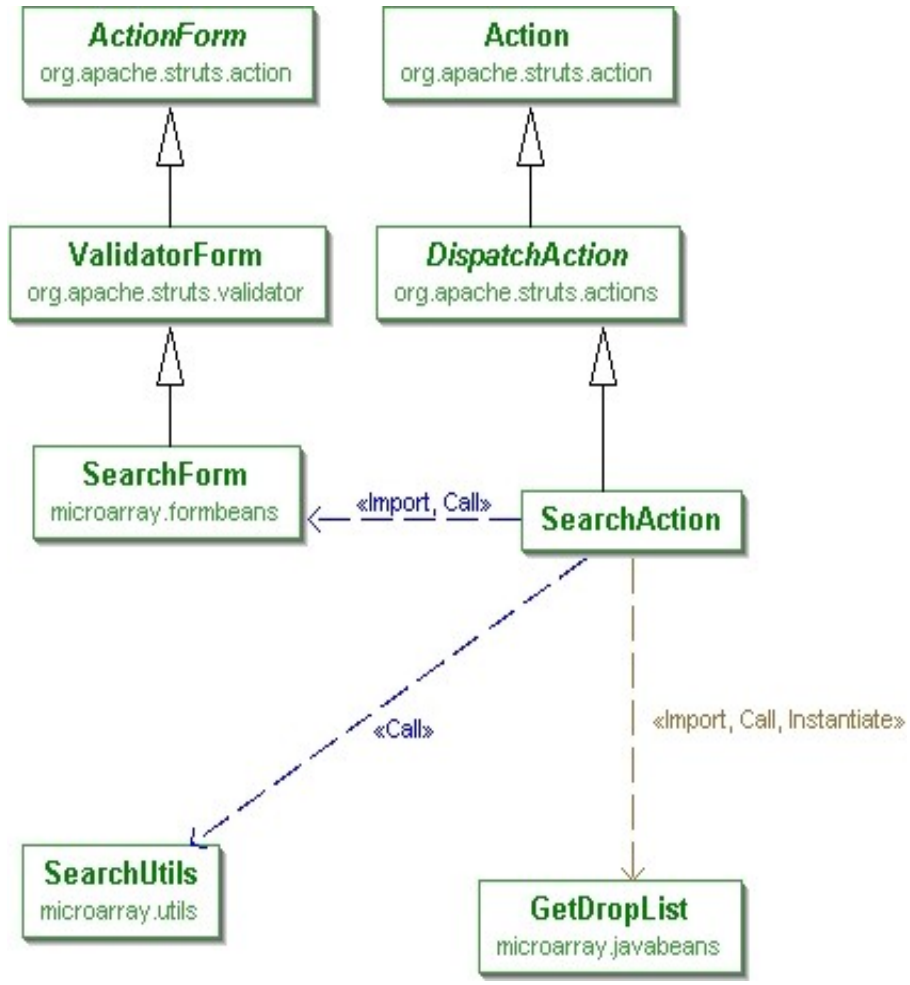

Figura 3.60.: Class diagram costruito attorno a JoinAction.

3.10.1.6 View platforms

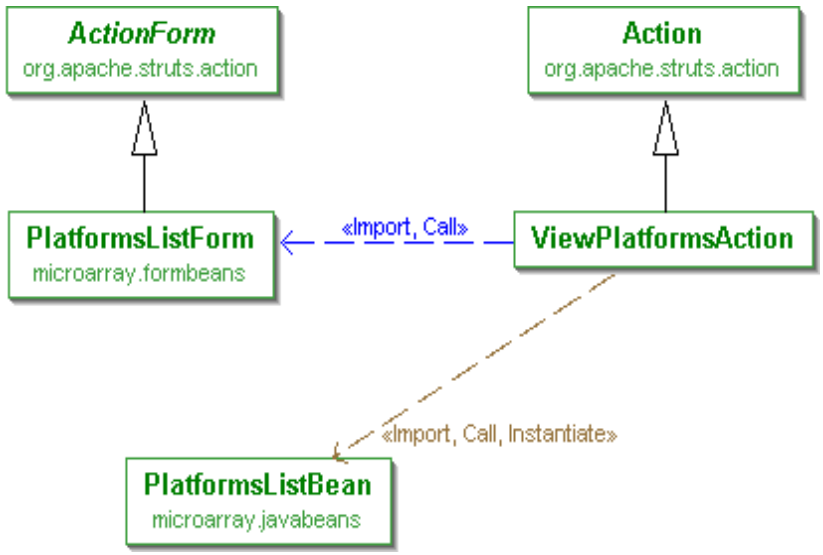

Figura 3.61.: Class diagram costruito attorno a ViewPlatformsAction.

3.10.1.7 Experiments administration

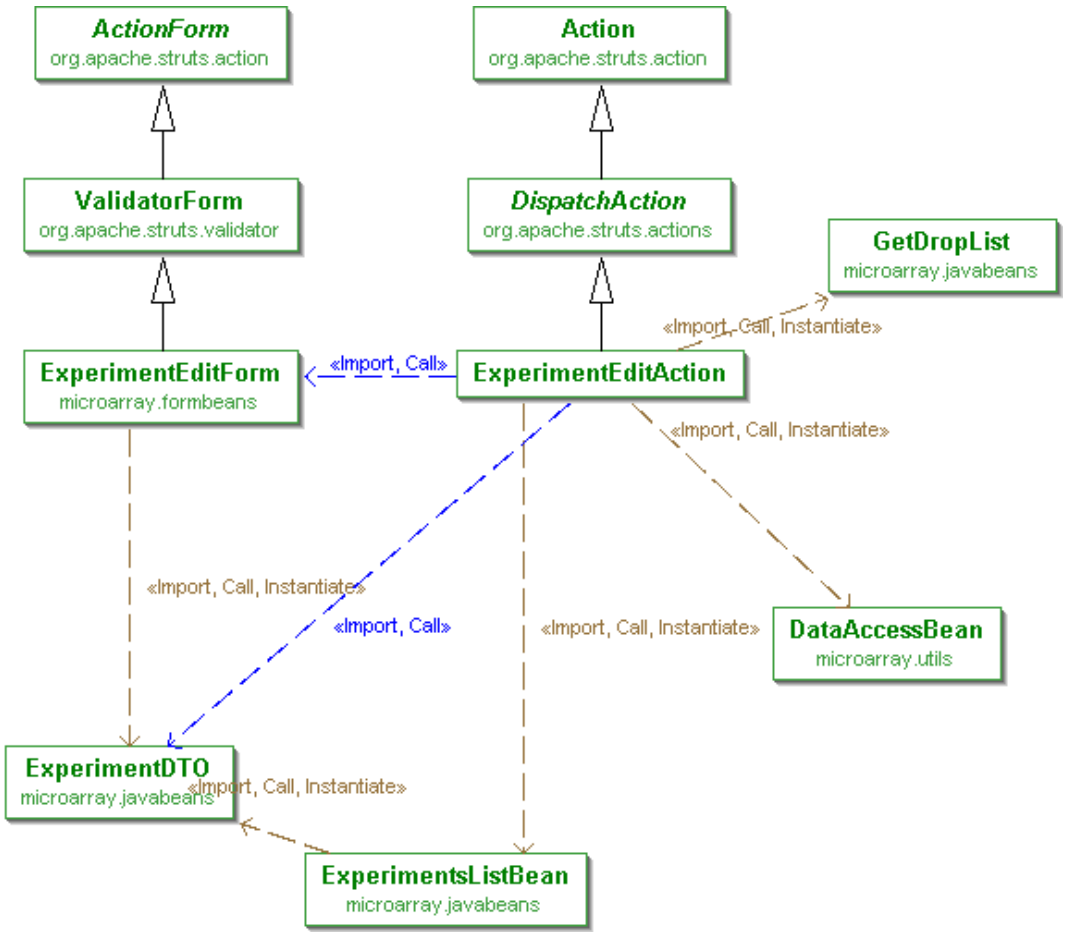

Figura 3.62.: Class diagram costruito attorno a ExperimentEditAction.

3.10.1.8 Users administration

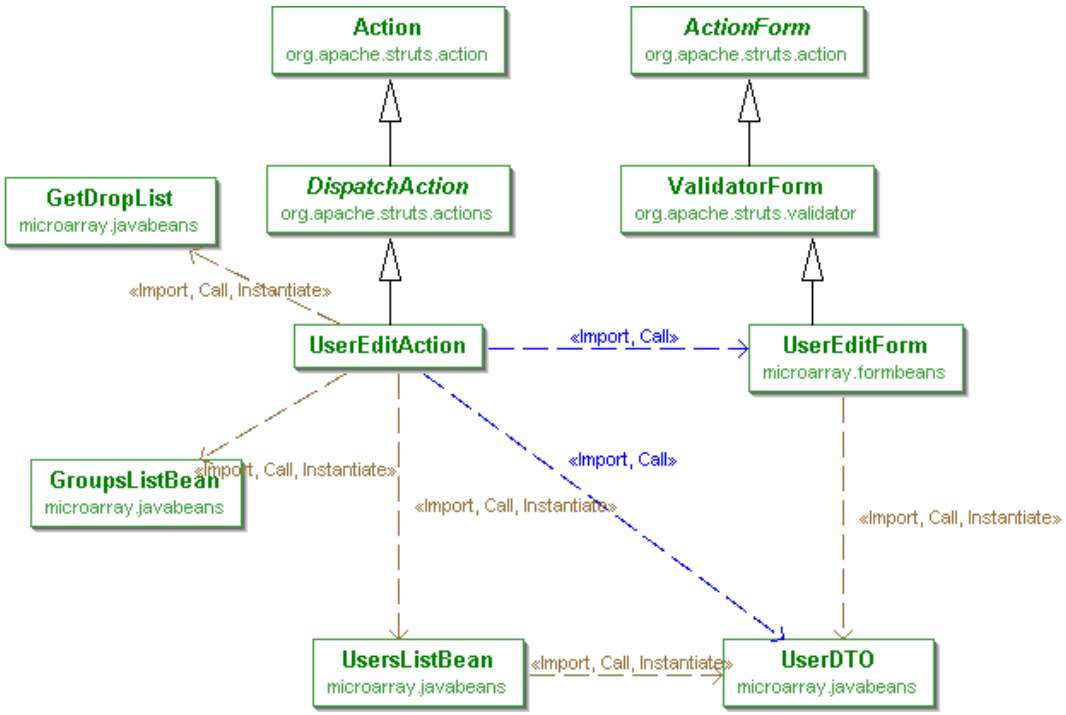

Figura 3.63.: Class diagram costruito attorno a UserEditAction.

3.10.1.9 Index administration

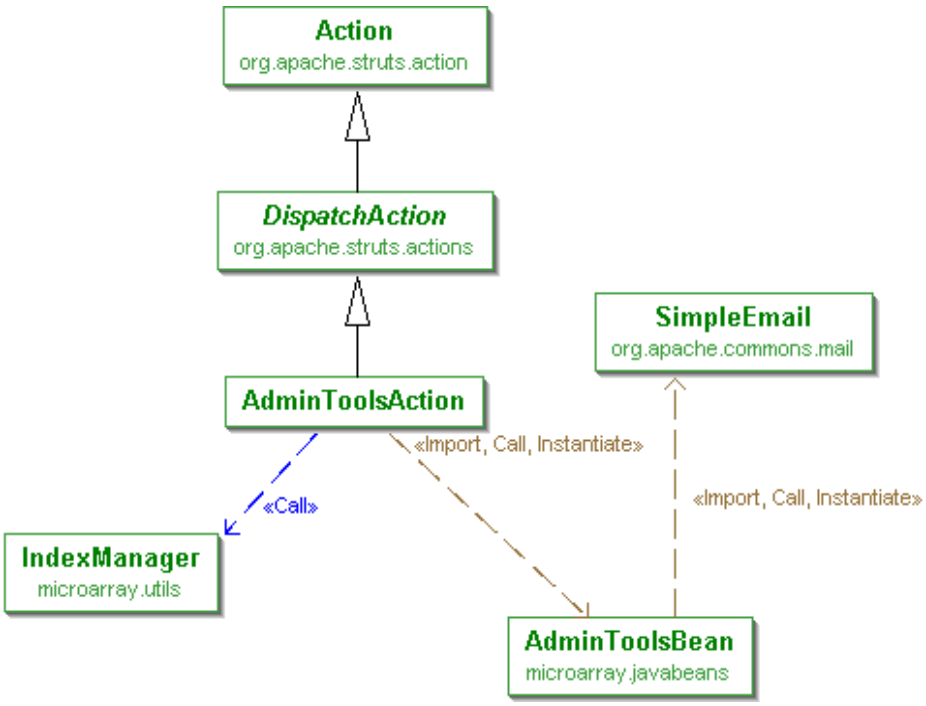

Figura 3.64.: Class diagram costruito attorno a AdminToolsAction.

4 Indicizzazione e ricerca

Contenuto del capitolo

4.1 Il problema dell'information retrieval

Prima di affrontare il problema dell'indicizzazione e della ricerca ci sembra opportuno introdurre il tema dell'information retrieval. Per information retrieval, o più sinteticamente IR, si intende generalmente la scienza della ricerca di informazioni all'interno di documenti, della ricerca dei documenti stessi o della ricerca tra i metadati che li descrivono. Si tratta di una scienza interdisciplinare, basata principalmente su discipline come l'informatica, la matematica, la biblioteconomia, la psicologia cognitiva, la linguistica e la statistica.

L'obbiettivo principale dell'information retrieval è la riduzione del cosiddetto “information overload” che consiste nell'avere troppe informazioni per poter prendere una decisione o per poter essere informato su un argomento. Infatti, quando le informazioni sono molto numerose, ridondanti, contraddittorie e, per di più, in continua e rapida crescita, individuare quali siano utili a prendere una decisione può diventare molto difficile.

I più comuni e diffusi sistemi per l'information retrieval sono sicuramente i motori di ricerca (search engines) nel campo dell'informatica. I motori di ricerca provano a ridurre gli effetti indesiderati dovuti all'information overload minimizzando il tempo necessario a trovare le informazioni e l'ammontare stesso delle informazioni da consultare. Tra di essi i più popolari sono sicuramente quelli legati alla ricerca sul Web, ma ne esistono di operanti all'interno delle intranet aziendali, sui personal computer e persino in ambiente mobile.

Tra gli ambienti in cui è più sentito il problema dell'information overload possiamo sicuramente inserire il mondo della ricerca sull'espressione genica che produce, oggi, un grande ammontare di dati. L'organizzazione, la catalogazione e l'archiviazione di tali dati

è una tematica ampiamente trattata a cui sono state date numerose soluzioni. Proliferano, infatti, i microarray database, basi di dati che immagazzinano i dati e li rendono disponibili agli utenti per il download o per l'analisi. La prima parte di questo nostro lavoro è un esempio di microarray database.

Raggiunto l'obbiettivo dell'archiviazione dei dati nasce un interrogativo: è possibile effettuare ricerche? La seconda parte di questo lavoro riguarda proprio la possibilità di effettuare ricerche sui dati. Si voleva dare all'utente del nostro sistema la possibilità di effettuare ricerche che non coinvolgessero semplicemente i metadati che descrivono gli esperimenti, come il titolo, la descrizione, i nomi degli organismi coinvolti e i trattamenti cui sono stati sottoposti, ma che si spingessero fin dentro i dati veri e propri per indagare sul comportamento dei singoli geni in diverse situazioni.

In sostanza il nostro principale obbiettivo è stato rispondere a domande del tipo: che valori di espressione sono stati ottenuti per il gene X negli esperimenti in cui è stato coinvolto? Come si sono comportati i geni accomunati dalla caratteristica Y negli esperimenti archiviati? Quale è stato il comportamento dei geni X, Y e Z nelle analisi effettuate?

Per ottenere risposte a simili domande bisognava trovare il modo di costruire un indice dei dati archiviati. Senza un indice ogni ricerca avrebbe implicato la scansione seriale del contenuto di ogni dato archiviato, il che avrebbe impiegato un tempo sicuramente incompatibile con l'utilizzo da parte di un operatore umano. L'utilizzo di un indice, invece, consente di ridurre i tempi di ricerca al prezzo, però, dell'utilizzo di maggiori risorse in termini di spazio su disco, per l'archiviazione, e di occupazione di CPU, per la creazione e l'aggiornamento.

La possibilità di cimentarsi nella progettazione e realizzazione di un sistema di indicizzazione “ad hoc” per i dati di espressione genica è stata subito scartata per via della complessità della materia,

sicuramente incompatibile con i tempi di un lavoro di tesi. E' stata esaminata, invece, la possibilità di utilizzare un sistema IR già realizzato, gratuito e possibilmente open source. Abbiamo quindi concentrato la nostra attenzione sul come rendere compatibili le informazioni da indicizzare con un sistema IR general-purpose.

4.2 I dati da indicizzare

Come detto in precedenza, gli esperimenti sono descritti da una serie di metadati scelti in accordo a quanto specificato dallo standard MIAME 1.1. Per quanto riguarda i dati veri e propri di cui constano gli esperimenti, invece, lo standard MIAME non impone l'utilizzo di un formato in particolare. Allo stato attuale i dati si presentano in una miriade di formati diversi, basati perlopiù sull'utilizzo di file di testo o di fogli elettronici.

Il formato dei dati utilizzato dai ricercatori è fortemente legato ai tool di analisi a corredo delle apparecchiature utilizzate per la sperimentazione. Comunque i ricercatori spesso esportano i dati per utilizzarli nei più comuni spreadsheet, Microsoft Excel in testa, approfittando del fatto che è sempre possibile una rappresentazione tabellare dei dati, in qualsiasi modo questi siano elaborati.

E' emerso, dalla fase di analisi dei requisiti, la necessità di archiviare, per ogni esperimento, tre diverse tipologie di dati:

- dati grezzi, un archivio compresso per ogni esperimento;
- dati normalizzati, un file di testo per ogni esperimento;
- liste di geni regolati (o selezionati), un file di testo per ogni analisi effettuata nel corso di un esperimento.

I dati grezzi sono l'output primario di ogni esperimento. Essendo dati non sottoposti ad alcuna elaborazione sono di difficile interpretazione e sovrabbondanti rispetto agli scopi della ricerca. Per questo motivo si è scelto di accettare la sottomissione di tali dati esclusivamente sotto forma di archivio compresso. Non ritenendo utile la ricerca su questi dati, non ne abbiamo previsto l'indicizzazione.

ProbeId	Symbol	O_0_17_35_B	O_0_17_51_B	O_0_17_52_A	O_3_17_35_D	O_3_17_51_D	O_3_17_52_C	O_6_17_56_C	O_6_17_57_C	O_6_17_57_G
1450041	**Dclre1c**	181	189	159	179	177	188	156	158	160
4540037	**Mdm1**	202	165	196	208	221	214	180	210	214
1780056	**Zp3r**	139	187	187	177	196	187	183	172	166
610408	**E430018M08Rik**	185	168	172	165	150	136	198	162	181
610369	**Ampd3**	184	181	143	163	153	144	199	147	198
1450014	**D830016O14Rik**	141	107	130	149	136	158	147	165	145
380019	**Crry**	318	342	309	308	347	334	431	479	318
6860707	**4930470D19Rik**	234	247	291	217	194	258	219	224	237
1850619	**Sfrs16**	148	113	155	162	144	141	134	147	149
3780279	**Prkcz**	681	702	640	428	449	433	339	362	333
610088	**2810028N01Rik**	159	165	190	160	155	181	179	194	199

Figura 4.1.: Un estratto da una tabella di dati normalizzati. Gli indici di colonna corrispondono a campioni e/o ibridazioni. Gli indici di riga, in questo caso probeid, identificano il probe dell'array. Nella seconda colonna il symbol è uno degli identificativi utilizzati per i geni nel mondo della ricerca. Conseguentemente la tabella contiene il valore id espressione rilevato su un probe nell'analisi di un campione e/o ibridazione.

I dati normalizzati vengono ricavati dall'elaborazione dei dati grezzi. Questi dati sono leggibili da un operatore umano, e per questo, già nella progettazione dell'applicazione, avevamo previsto la possibilità di inserirli e scaricarli nel formato di file di testo. Un file di dati normalizzati consta in una tabella i cui indici di colonna sono campioni e/o ibridazioni e i cui indici di riga sono gli identificatori di probe o gruppi di probe dell'array. Al centro della tabella, quindi, c'è il valore di espressione, misurato in corrispondenza di un probe, di un gene appartenente ad un campione e/o ibridazione analizzato. Di tali dati abbiamo ritenuto necessaria l'indicizzazione.

Ad ogni analisi effettuata sui dati di un esperimento corrisponde un file di geni regolati, o selezionati. Questi dati, oltre ad essere stati oggetto di ulteriori elaborazioni, spesso vengono scremati delle informazioni inutili e, per questo, hanno una leggibilità maggiore e una dimensione minore. Le liste dei geni regolati sono organizzate in tabelle con formato analogo a quello visto per i dati normalizzati. Si era già previsto, nella fase di progettazione del sottosistema di archiviazione, l'inserimento sotto forma di file di testo e il download nello stesso formato: ora abbiamo ritenuto necessaria anche l'indicizzazione.

A questi dati si aggiungono le tabelle delle annotazioni di ogni piattaforma utilizzata. Dalla tabella delle annotazioni è possibile risalire ad informazioni sul gene, o sul gruppo di geni, legati ad ogni probe. E' quindi indispensabile, nella la lettura dei dati di ogni esperimento o analisi, potersi collegare alla tabella delle annotazioni del chip utilizzato(piattaforma nel nostro modello concettuale). Il contenuto di una tabella delle annotazioni è organizzato in modo tabellare con indici di riga i probe dell'array. Ogni colonna contiene metadati descrittivi del gene cui è collegato il probe, inclusa una ontologia, una descrizione, il nome standard, i sinonimi, un identificativo univoco ecc.

TargetID	ProbeID	ONTOLOGY_COMPONENT
15E1.2	20605	cellular component unknown [goid 8372] [evidence ND]
ABCA2	3520743	integral to membrane [goid 16021] [evidence NAS]; ATP-binding cassette (ABC) transporter complex [goid 43190] [pmid 11178988] [evidence NAS]; lysosomal membrane [goid 5765] [pmid 11309290] [evidence IDA]
76P	3060450	centrosome [goid 5813] [pmid 10562286] [evidence TAS]; microtubule [goid 5874] [evidence IEA]; gamma-tubulin ring complex [goid 8274] [pmid 10562286] [evidence NAS]
AADAC	580711	endoplasmic reticulum membrane [goid 5789] [pmid 15152005] [evidence IDA]; membrane [goid 16020] [evidence IEA]; microsome [goid 5792] [pmid 8063807] [evidence TAS]; integral to membrane [goid 16021] [evidence IEA]
A4GNT	4590047	membrane fraction [goid 5624] [pmid 10430883] [evidence TAS]; integral to membrane [goid 16021] [pmid 10430883] [evidence TAS]; Golgi stack [goid 5795] [evidence IEA]
A2BP1	770300	Golgi apparatus [goid 5794] [pmid 10814712] [evidence TAS]
A2BP1	3290546	Golgi apparatus [goid 5794] [pmid 10814712] [evidence TAS]
A2BP1	3420601	Golgi apparatus [goid 5794] [pmid 10814712] [evidence TAS]
A2M	7400044	extracellular region [goid 5576] [pmid 14718574] [evidence NAS]
A4GALT	5670674	membrane fraction [goid 5624] [pmid 10748143] [evidence IDA]; integral to Golgi membrane [goid 30173] [pmid 10748143] [evidence NAS]; integral to membrane [goid 16021] [evidence IEA]; Golgi stack [goid 5795] [evidence IEA]; membrane [goid 16020] [evidence IEA]

Figura 4.2.: Estratto da una tabella delle annotazioni. Il probeid identifica univocamente i probe dell'array. Per ogni probe è riportata una serie di informazioni riguardanti il gene ad esso collegato, in questo caso è mostrata una ontologia.

Un punto fondamentale per l'evoluzione del nostro lavoro è stato sicuramente l'assunto che dai dati degli esperimenti, memorizzati in forma tabellare in uno spreadsheet, fosse sempre possibile trarre un file di testo tabulato e viceversa. Quasi tutti gli editor per fogli di calcolo, infatti, offrono supporto all'esportazione dei propri spreadsheet sotto forma di file di testo tabulati.

La natura testuale e tabellare dei dati da indicizzare ha ristretto la nostra ricerca di un sistema per l'information retrieval open source al campo dei soli motori di ricerca testuali. Bisognava però tenere conto del fatto che, mentre un motore di ricerca general purpose, è pensato per l'indicizzazione e la ricerca su documenti relativamente brevi, quali possono essere notizie d'agenzia, i post di un forum o pagine web, i documenti oggetto del nostro sistema sono tabelle di decine di migliaia di righe e centinaia di colonne che raggiungono spesso dimensioni dell'ordine dei 20Mbyte.

Per questo motivo in un motore di ricerca abbiamo ricercato, prima di tutto, la flessibilità, con la speranza di trovare una soluzione che consentisse la realizzazione delle funzionalità di ricerca anche su dati così insoliti per il settore.

4.3 Analisi delle tecnologie IR disponibili

Le tecnologie per l'information retrieval disponibili nel mondo del freeware e/o open source sono molto numerose, e costituiscono un mondo in continua evoluzione e trasformazione. Come sempre accade nel mondo dell'open source, nuovi progetti nascono, evolvono, si fermano o muoiono con gran velocità e in modo imprevedibile. Nell'impossibilità di comparare un numero tanto grande di tecnologie, abbiamo concentrato la nostra attenzione sui prodotti più popolari, più diffusi e meglio documentati e supportati. In particolare, sono state prese in esame e confrontate due soluzioni:

- la funzionalità di ricerca “full text” integrata in MySQL;
- l'utilizzo di un sistema IR stand alone, Apache Lucene.

La prima soluzione, pur non essendo nota per le performance particolarmente brillanti, è stata esaminata perchè integrata in uno strumento già utilizzato, MySQL appunto. Apache Lucene, invece, è stato preso in esame per le ottime credenziali offerte:

- l'essere un prodotto open source di Apache Software Foundation;
- i numerosi port verso linguaggi come C, C++, Delphi, Perl, Pyton, C#.NET, PHP e Ruby;
- l'utilizzo in celebri prodotti tra cui MediaWiki, Eclipse, FastFind, Nabble.

La prima delle credenziali indicate, l'essere un progetto Apache, pur non essendo ricollegabile direttamente alla qualità del prodotto, ci offre una garanzia sull'aspettativa di vita del progetto Lucene, fondamentale per la futura evoluzione del sistema a cui stiamo lavorando. Le altre due credenziali, i port verso altri linguaggi e l'utilizzo in noti prodotti, invece ci offrono una misura indiretta di quanto Lucene sia vicino allo stato dell'arte nel mondo dell'IR.

Non meno importante è il fatto che le due tecnologie analizzate si adattino bene alla piattaforma di sviluppo utilizzata. Se da un lato Apache Lucene è un progetto disponibile su piattaforma Java, la stessa utilizzata nello sviluppo del nostro sistema, dall'altro lato la funzionalità di ricerca fulltext in MySQL è addirittura completamente indipendente dalle tecnologie utilizzate nello sviluppo del software. In sostanza, i due prodotti analizzati erano anche quelli che minimizzavano i problemi di interoperabilità tra piattaforme.

4.3.1 MySQL fulltext

Su tutte le colonne di testo presenti nelle tabelle di MySQL 5.0, cioè sulle colonne di tipo CHAR, VARCHAR TEXT o LONGTEXT, è possibile attivare la funzionalità di ricerca fulltext. Per l'attivazione di tale funzionalità è necessaria la creazione di un indice fulltext associato alla colonna. La definizione di un indice FULLTEXT può essere inserita nello statement CREATE TABLE, al momento della creazione di una tabella, o aggiunta più tardi utilizzando uno statement ALTER TABLE o CREATE INDEX, su una tabella esistente.

Una volta creato l'indice è possibile effettuare ricerche utilizzando una sintassi del tipo MATCH...AGAINST nella clausola WHERE. Alla funzione MATCH() vanno passati i nomi delle colonne su cui si vuole effettuare la ricerca; alla funzione AGAINST() va passato il testo che si vuole ricercare e, opzionalmente, un modifier che indica la tipologia di ricerca da effettuare. Per ogni riga tra quelle che gli sono passate, la funzione MATCH() restituisce un punteggio che misura la somiglianza tra la stringa ricercata e il testo presente nella riga stessa. Una semplice query di ricerca fulltext potrebbe essere:

```
SELECT * FROM articles
WHERE MATCH (title,body) AGAINST ('database');
```

che chiaramente effettua la ricerca fulltext della parola “database” nelle colonne “title” e “body” della tabella “articles”.

Per mezzo del modifier passato alla funzione AGAINST() è possibile scegliere una delle modalità di ricerca supportate da MySQL:

- ricerca in linguaggio naturale che interpreti le parole costituenti la stringa di ricerca come una normale frase;
- ricerca booleana che utilizzi operatori booleani in aggiunta alle parole da cercare;
- ricerca con espansione della query, che modifichi una stringa inserita in linguaggio naturale introducendo nuove parole scelte in base a regole di rilevanza.

E' bene puntualizzare che non bisogna confondere la funzionalità di ricerca fulltext con l'utilizzo dell'operatore LIKE che, pur potendo restituire potenzialmente gli stessi risultati, non utilizzando un indice, non è in grado di ordinare i risultati per rilevanza. L'operatore LIKE, infatti, si limita a fare il parsing di determinati contenuti alla ricerca della chiave che gli viene passata come parametro; quindi, se trova la stringa cercata, restituisce in output tutti i record che la contengono. Ciò comporta l'assoluta incapacità dell'operatore LIKE di scalare alla crescita del numero di record archiviati.

4.3.2 Apache Lucene

Lucene consta in una complessa API per l'information retrieval scritta interamente in Java. Le funzionalità offerte dall'API sono essenzialmente due: indicizzazione e ricerca di testo. Dato un insieme di documenti di testo, Lucene può costruire un indice per poi consentire di effettuarci sopra delle ricerche. Al centro dell'architettura di Lucene c'è il concetto di documento come insieme di campi testuali.

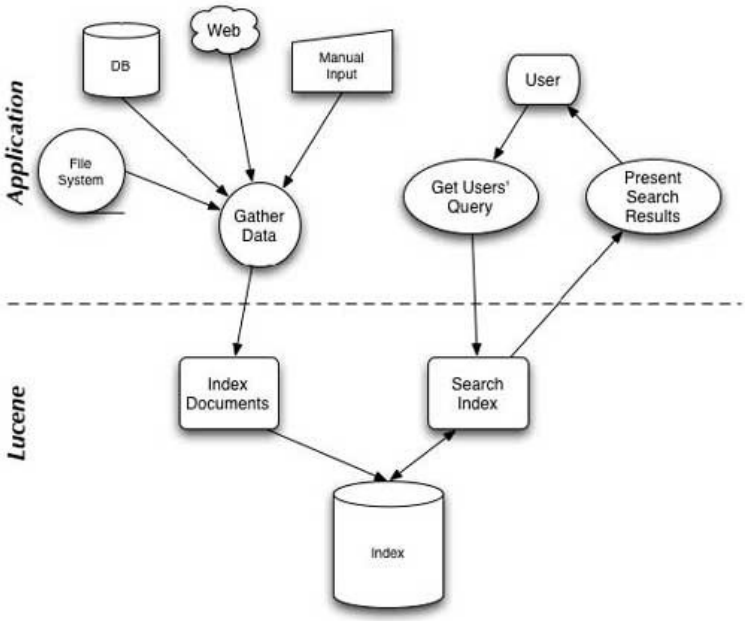

Figura 4.3.: Lucene offre solo le funzionalità per l'indicizzazione e la ricerca. Il resto deve essere gestito a livello applicativo.

Ogni documento (Document nel linguaggio di Lucene) è costituito da un numero variabile di campi testuali (Field) formati da una coppia <nome,contenuto>. Sia il campo nome che il campo contenuto di ogni Field assumono solo ed esclusivamente valore testuale. Ogni qual volta si voglia indicizzare un dato tramite Lucene occorre convertirlo in un Document costituito da un insieme di Field. Ciò rende molto flessibile l'utilizzo di Lucene, che può indicizzare indifferentemente file di testo, file html, documenti di Ms Word, pdf, spreadsheet e qualunque altro formato di file da cui sia possibile ricavare un testo.

Lucene, inoltre, consente di specificare, per ogni Field le operazioni da effettuare su di esso rispetto all'indicizzazione e all'archiviazione. L'indicizzazione, come è intuibile, riguarda l'analisi del testo, l'individuazione dei token, e l'aggiunta nell'indice di un riferimento al Document in corrispondenza dei token individuati. L'archiviazione invece riguarda la possibilità di archiviare, in una struttura dati esterna all'indice, ma sempre gestita da Lucene, il contenuto originale del Field, cosa assai utile, ad esempio, per memorizzare gli identificativi dei dati nelle tabelle di un database relazionale.

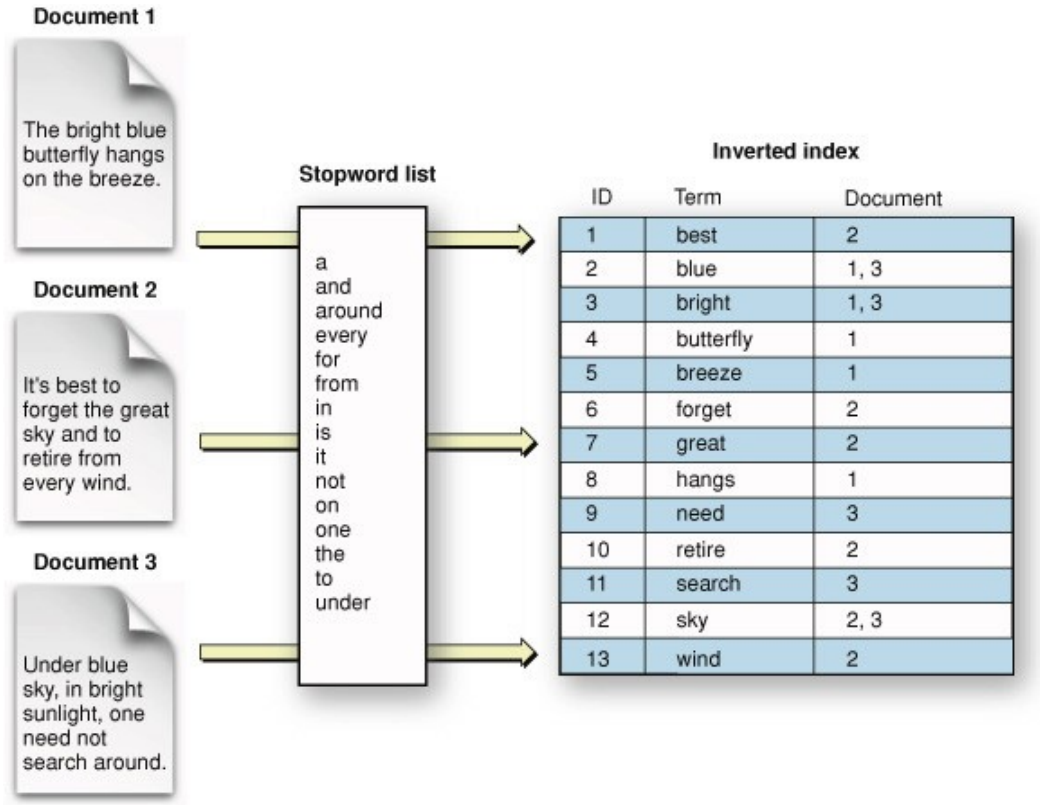

Figura 4.5.: Semplice esempio di inverted index creato a partire da documenti il cui contenuto è filtrato da una stop-words list.

A partire dai documenti Lucene crea un indice, normalmente contenuto in una cartella del file system, ma che può essere indifferentemente memorizzato nella memoria RAM o perfino distribuito su rete. L'indice di Lucene è strutturato come uno “inverted index”, un indice “capovolto” in cui a partire dal termine indicizzato, si risale alla lista di tutti i documenti che lo contengono. Una esemplificazione di come funziona un inverted index è riportato in figura 4.5.

Riassumiamo i principali punti di forza di Lucene nei punti seguenti:

- la possibilità di gestire una indicizzazione incrementale o di tipo batch;

- la possibilità di indicizzare qualunque tipo di dato, purché esista una API che consenta la lettura del contenuto testuale in esso contenuto;
- il pieno controllo del processo di indicizzazione, delegando all'applicazione il compito di raccogliere e trovare i documenti;
- la possibilità di restringere la ricerca ad un particolare Field presente nel Document;
- il supporto all'eliminazione delle stop-words, come ad esempio le inglesi "a", "and", "the", “this”;
- lo stemming, col quale è possibile estendere la ricerca di una parola a tutte le parole accomunate dalla stessa radice;
- l'accesso concorrente all'indice, anche quando questo è in fase di update, grazie ad un sistema di read e write lock;
- il supporto a molte lingue diverse dall'inglese;
- la possibilità di effettuare, dall'API, query booleane, query con wildcards, range query e soprattutto query in linguaggio naturale;
- la possibilità di applicare filtri ai risultati della ricerca;
- l'ordinamento autmatico dei risultati in base al valore di uno score;
- le capacità di ricercare più parole ordinando i risultati in base a quanto queste sono vicine tra loro (proximity query) e la capacità di cercare varianti delle parole (fuzzy query);
- il caching delle query effettuate, che consente uno speed up delle query più frequenti;
- il term boosting, col quale è possibile associare un coefficiente di rilevanza ai documenti contenenti un certo termine;
- il supporto all'highlighting dei risultati della ricerca.

A quanto sopra va aggiunta la possibilità di indicizzare Field contenenti valori da trattare come keyword, cioè da non sottoporre al processo di analisi e tokenizzazione del testo. Questa tipologia di Field consente di risalire dal Document indicizzato da Lucene al dato comunque esso sia archiviato, ad esempio consente di risalire

ad un dato contenuto in una tabella di un database relazionale, tramite il valore di una chiave primaria, o ad un dato archiviato nel file system, tramite un path. Ne deriva una ulteriore flessibilità di Lucene, stavolta rispetto alla modalità di archiviazione dei dati.

4.3.3 MySQL fulltext Vs Lucene

Nel confrontare le due tecnologie abbiamo da un lato la complessità e le potenzialità di Lucene, dall'altro la semplicità di utilizzo della ricerca fulltext in MySQL.

Come abbiamo visto l'utilizzo della ricerca fulltext in MySQL è piuttosto semplice: basta creare un indice FULLTEXT su una colonna di tipo testuale e poi effettuare query utilizzando le funzioni MATCH(), AGAINST(). MySQL gestisce internamente l'aggiunta e la modifica di documenti all'indice contestualmente all'INSERT o UPDATE dei dati contenuti nelle tabelle. Dall'altro lato l'aggiunta/modifica di documenti all'indice di Lucene va gestita programmaticamente tramite l'API. Da un lato le query possono essere effettuate, ad esempio attraverso il driver JDBC già utilizzato dal nostro applicativo, dall'altro lato, invece, occorre utilizzare la API di Lucene e gestire l'indice sul file system attraverso quella.

Lucene, grazie all'utilizzo di un inverted index, è molto veloce ad effettuare query su indici contenenti un gran numero di documenti, mentre, dal lato di MySQL la velocità scende all'aumentare del numero di documenti. MySQL, infatti, utilizzando la RAM per il caching sia dei risultati delle query che dell'indice, e ciò può essere causa di frequenti swap al crescere del numero dei dati da gestire.

Un rallentamento in corrispondenza degli INSERT su tabelle indicizzate è inevitabile in MySQL. Ad ogni inserimento, il processo di indicizzazione comporta un rallentamento nell'operazione. Dopo che un nuovo documento è stato indicizzato MySQL deve aggiornare la propria copia dell'indice in RAM con quella nuova: ciò causa un

rallentamento quando si effettuano contemporaneamente inserimenti, aggiornamenti e ricerche. Dall'altro lato Lucene consente di scegliere se differire l'indicizzazione, o se effettuarla al volo, e inoltre non ha la necessità di sostituire l'intero indice ad ogni nuova indicizzazione. Bisogna comunque notare come l'update dell'indice, che in MySQL è automatico, in Lucene non è previsto. Aggiornare l'indice di Lucene vuol dire eliminare la vecchia versione del documento e poi indicizzare la nuova.

Lucene offre un controllo maggiore sul processo di analisi del testo. L'eliminazione delle stop-words, la tokenizzazione, e la conversione in minuscolo dei token, che in MySQL sono automatiche, possono essere abilitate o disabilitate a piacere con Lucene, offrendo così una maggiore flessibilità nel trattamento dei documenti. In questo modo, utilizzando Lucene, è possibile effetture query case sensitive, cosa impossibile con MySQL.

Per quanto riguarda la complessità delle query realizzabili, MySQL è molto meno dotato, infatti non prevede proximity query, wildcard query, fuzzy query e il term boosting. A vantaggio di MySQL, invece, è la possibilità di effettuare il join di due indici utilizzando la comune sintassi SQL; cosa non prevista in Lucene.

Analizzate le caratteristiche dell'una e dell'altra soluzione, la nostra scelta è ricaduta sul prodotto Apache. Le motivazioni che hanno portato a scegliere Lucene sono riassumibili nei seguenti punti:

- l'utilizzo di Lucene consente di tenere separato il sistema di indicizzazione e di gestione dell'indice dal sistema di archiviazione dei dati stessi già creato nella prima fase;

- Lucene offre un più ampio ventaglio di scelte sul come attuare l'indicizzazione;
- la possibilità di sfruttare contemporaneamente la velocità di MySQL come RDBMS e la velocità di Lucene come motore di ricerca tenendo separate le performance dell'uno e dell'altro;
- la possibilità di attuare una indicizzazione batch in orari di scarso carico del sistema;
- la possibilità di costruire query complesse programmaticamente a partire dalle ricerche in linguaggio naturale dell'utente.

In questa fase del processo di scelta è stato possibile prevedere, come principale svantaggio di una soluzione ibrida, la necessità di gestire l'allineamento del contenuto del database al contenuto dell'indice. Le ulteriori difficoltà implementative cui siamo andati incontro scegliendo l'utilizzo di Lucene sono evidenti da quanto già affermato nella comparazione delle due tecnologie.

4.4 Progettazione del sottosistema di indicizzazione e ricerca

Effettuare una progettazione a valle di una scelta tecnologica potrebbe apparire insolito e poco professionale. In realtà, però, bisogna tenere conto di un fatto fondamentale: il mondo dell'information retrieval è tutt'altro che standardizzato. Già nella comparazione della funzionalità di ricerca fulltext di MySQL a Apache Lucene dovrebbero essere emerse la qualità e la quantità delle differenze tra i due approcci all'IR. Modi tanto diversi di gestire i documenti, gli indici e di accedere ai contenuti, ci hanno portato alla conclusione che solo a partire da una conoscenza approfondita delle caratteristiche del sistema IR, potevano scaturire scelte progettuali.

Per i motivi di cui sopra, una volta scelta la tecnologia, Apache Lucene, abbiamo dato inizio alla progettazione del nostro sottosistema per la ricerca e siamo partiti, come è ovvio, dalla raccolta e analisi dei requisiti. Abbiamo poi cercato di mappare i nostri requisiti sulle funzionalità offerte da Lucene, compatibilmente con le risorse a nostra disposizione, per ottenere quanto desiderato dal committente.

4.4.1 Raccolta e analisi dei requisiti

Dal contatto con il committente, i bioinformatici del CNR di Avellino, sono emersi requisiti solo embrionali per le funzionalità di ricerca. I nostri committenti, infatti, non erano a conoscenza, né lo sono tuttora, dell'esistenza di sistemi che gestissero la ricerca tra gli esperimenti di espressione genica e quindi, non avevano un termine di paragone a partire dal quale esprimere le proprie necessità.

Mancando di un solido riferimento, i requisiti espressi per la ricerca si concentravano a poche funzionalità, la risposta a poche domande cui i sistemi già noti non erano in grado di rispondere. Per questo motivo i requisiti iniziali sono stati integrati nel corso del lavoro di sviluppo a partire dalle funzionalità già ottenute, confrontandosi col committente.

Per ricerca il committente intende genericamente la possibilità di cercare, contemporaneamente, nei dati di tutti gli esperimenti archiviati, quello che normalmente può essere cercato utilizzando un qualsiasi editor in un solo dato. Tramite la ricerca si vuole in pratica, ottenere la possibilità di confrontare il comportamento di un gene in diversi esperimenti, vale a dire in diversi contesti, dopo aver subito diversi trattamenti.

Come è noto, ogni editor di testo o fogli di calcolo, è in grado di effettuare ricerche sui documenti, tramite scansione sequenziale degli stessi. I prodotti più recenti consentono anche di estendere la ricerca ai documenti contenuti in una cartella, o ai documenti contenuti sull'intero disco di un PC. Quindi, qualora si voglia cercare, ad esempio, l'identificatore di un gene, il nostro editor di testo comincerà la scansione sequenziale dei documenti e aprirà uno ad uno tutti i documenti che gli sono offerti. Al termine della ricerca avremo effettuato la scansione di un gran numero di documenti, ognuno di essi sarà stato aperto dall'editor, e all'utente resterà di navigare da un risultato all'altro alla ricerca di quanto interessa.

E facile intuire come simile operazione, effettuata su decine, magari centinaia di file di circa 10Mbyte, richieda una quantità spropositata di risorse computazionali e impieghi un tempo incompatibile col lavoro di un ricercatore. Comunque noi, invece che concentrare la nostra attenzione sull'enorme quantità di risorse computazionali richieste da tale operazione, vogliamo concentrare l'attenzione sulla qualità del risultato della ricerca, perché questo è il problema da risolvere.

L'utilizzo di un motore di ricerca consente, infatti, un sicuro abbattimento delle risorse computazionali necessarie ad effettuare la ricerca: nessuna scansione sequenziale è richiesta qualora si siano indicizzati i dati. Dall'altro lato, dal lato della qualità dei risultati della ricerca, però, solo una corretta progettazione dell'indice può fornire i risultati desiderati.

Il committente vuole poter interrogare il sistema chiedendo, ad esempio: quale esperimento coinvolge il gene X? Si aspetta che il sistema gli dia modo di accedere a tutti gli esperimenti archiviati che hanno interessato quel gene per visionarne il comportamento. Allo stesso modo il committente vuole poter cercare tra gli esperimenti il comportamento di un insieme definito di geni, specificandone gli identificatori, o di una categoria di geni, specificandone il nome, o dei geni accomunati da una particolare caratteristica, o coinvolti in un particolare processo, sempre semplicemente inserendo un testo.

Data la vastità di ogni file di dati, dati normalizzati o lista di geni selezionati, si è individuato, come requisito fondamentale, la capacità di offrire, come risultato della ricerca, un estratto di tali file; possibilmente solo la riga corrispondente ai valori di espressione dei geni cercati. Siccome ogni riga di una tabella non ha senso se non si conosce l'intestazione, il sistema deve anche fornire, come risultato della ricerca, l'intestazione della tabella da cui provengono i dati.

Poiché ogni piattaforma, cioè ogni chip, è costruito per scopi diversi e destinato a diversi esperimenti su diversi organismi, è necessario prevedere la possibilità di restringere il campo della ricerca ai soli esperimenti realizzati su una particolare piattaforma. Ciò anche alla luce della possibilità che probe di piattaforme diverse siano caratterizzati dal medesimo identificatore. Inoltre è ritenuta utile la possibilità di restringere la ricerca ai soli dati normalizzati o alle sole liste di geni regolati.

Altro requisito fondamentale, è la capacità di offrire, a corredo di ogni risultato della ricerca, un link all'esperimento o analisi di provenienza in modo che l'utente, quando interessato, possa risalire a maggiori dettagli sull'esperimento, sui campioni coinvolti e sulle analisi effettuate.

Un ultimo requisito, implicito, per via di quanto già sviluppato, sta nel garantire la riservatezza dei dati. Ogni utente, nell'effettuare la ricerca, deve poter accedere solo ai dati di esperimenti presenti nel suo ambito di visibilità. Per ulteriori dettagli riguardo alla definizione degli ambiti di visibilità si rimanda ai capitoli precedenti.

4.4.2 Use cases

Dai requisiti individuati per il sottosistema di indicizzazione e ricerca sono scaturiti nuovi dettagli che ci consentono di integrare il caso d'uso generale presentato per la ricerca nel capitolo sulla progettazione dell'applicazione. Questa volta, accanto all'interazione dell'utente col sistema, ci sarà una ulteriore interazione del sistema con l'indice, per mezzo dell'API di Lucene.

Per quanto riguarda, invece, l'amministrazione dell'indice e l'indicizzazione dei dati non riteniamo necessario specificare ulteriori dettagli rispetto a quanto già affermato.

4.4.2.1 Search

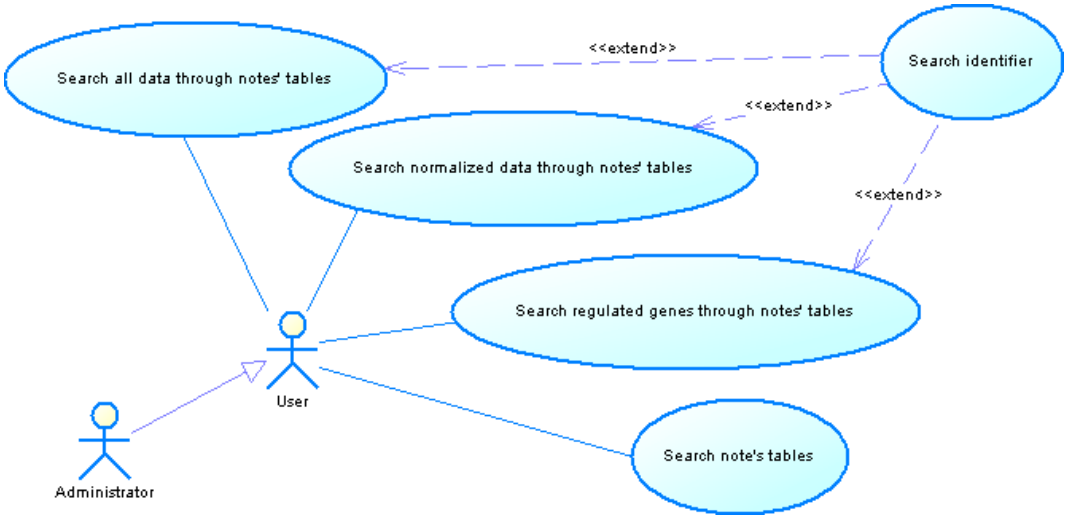

Figura 4.6.: Vista di dettaglio dello use case "Search".

4.4.2.2 Search all data through note's tables

Nome	Search all data through note's tables
Iniziatore	User/Administrator
Scopo	Effettuare una ricerca estesa alla totalità dei dati indicizzati
Precondizioni	Nessuna

Descrizione	1. L'utente chiede di effettuare una ricerca 2. Il sistema mostra all'utente la pagina contenente il form per la ricerca 3. L'utente inserisce il testo da ricercare, definendo lo scope della ricerca e opzionalmente la piattaforma interessata 4. L'utente trasmette i dati 5. Il sistema ricerca il testo inserito dall'utente nelle tabelle delle annotazioni 6. Lucene restituisce una collezione degli identificativi corrispondenti ai geni risultato della ricerca 7. Il sistema effettua una nuova ricerca sui dati normalizzati e sulle liste di geni selezionati 8. Lucene restituisce una collezione delle righe relative agli identificativi cercati 9. Il sistema presenta all'utente i risultati della ricerca
Alternative	1. Il sistema riconosce un testo illegale e chiede all'utente di effettuare una ricerca con un nuovo testo 2. Il sistema non trova una corrispondenza nell'indice e notifica all'utente l'assenza di risultati della ricerca
Postcondizioni	Nessuna

4.4.2.3 Search normalized data through note's tables

Nome	Search normalized data through note's tables
Iniziatore	User/Administrator
Scopo	Effettuare una ricerca estesa ai soli dati normalizzati
Precondizioni	Nessuna
Descrizione	1. L'utente chiede di effettuare una ricerca 2. Il sistema mostra all'utente la pagina contenente il form per la ricerca 3. L'utente inserisce il testo da ricercare, definendo lo scope della ricerca e opzionalmente la piattaforma interessata 4. L'utente trasmette i dati

	5. Il sistema ricerca il testo inserito dall'utente nelle tabelle delle annotazioni 6. Lucene restituisce una collezione degli identificativi corrispondenti ai geni risultato della ricerca 7. Il sistema effettua una nuova ricerca sui dati normalizzati 8. Lucene restituisce una collezione delle righe relative agli identificativi cercati 9. Il sistema presenta all'utente i risultati della ricerca
Alternative	1. Il sistema riconosce un testo illegale e chiede all'utente di effettuare una ricerca con un nuovo testo 2. Il sistema non trova una corrispondenza nell'indice e notifica all'utente l'assenza di risultati della ricerca
Postcondizioni	Nessuna

4.4.2.4 Search regulates genes through note's tables

Nome	Search regulated genes through note's tables
Iniziatore	User/Administrator
Scopo	Effettuare una ricerca estesa alle sole liste di geni selezionati
Precondizioni	Nessuna
Descrizione	1. L'utente chiede di effettuare una ricerca 2. Il sistema mostra all'utente la pagina contenente il form per la ricerca 3. L'utente inserisce il testo da ricercare, definendo lo scope della ricerca e opzionalmente la piattaforma interessata 4. L'utente trasmette i dati 5. Il sistema ricerca il testo inserito dall'utente nelle tabelle delle annotazioni 6. Lucene restituisce una collezione degli identificativi corrispondenti ai geni risultato della ricerca 7. Il sistema effettua una nuova ricerca sulle liste di geni selezionati

	8. Lucene restituisce una collezione delle righe relative agli identificativi cercati 9. Il sistema presenta all'utente i risultati della ricerca
Alternative	1. Il sistema riconosce un testo illegale e chiede all'utente di effettuare una ricerca con un nuovo testo 2. Il sistema non trova una corrispondenza nell'indice e notifica all'utente l'assenza di risultati della ricerca
Postcondizioni	Nessuna

4.4.2.5 Search note's tables

Nome	Search note's tables
Iniziatore	User/Administrator
Scopo	Effettuare una ricerca sulle sole tabelle delle annotazioni
Precondizioni	Nessuna
Descrizione	1. L'utente chiede di effettuare una ricerca 2. Il sistema mostra all'utente la pagina contenente il form per la ricerca 3. L'utente inserisce il testo da ricercare, definendo lo scope della ricerca e opzionalmente la piattaforma interessata 4. L'utente trasmette i dati 5. Il sistema ricerca il testo inserito dall'utente nelle tabelle delle annotazioni 6. Lucene restituisce una collezione delle righe relative relative al testo cercato 7. Il sistema presenta all'utente i risultati della ricerca
Alternative	1. Il sistema riconosce un testo illegale e chiede all'utente di effettuare una ricerca con un nuovo testo 2. Il sistema non trova una corrispondenza nell'indice e notifica all'utente l'assenza di risultati della ricerca
Postcondizioni	Nessuna

4.4.2.6 Search identifier

Nome	Search identifier
Iniziatore	User/Administrator
Scopo	Cercare nelle tabelle delle annotazioni i dettagli riguardo ad un gene di cui si conosce l'identificatore, probeid o targetid
Precondizioni	Il sistema ha precedentemente mostrato all'utente una pagina contenente i risultati di una ricerca sui dati
Descrizione	1. L'utente chiede di conoscere i dettagli del gene corrispondente ad un identificatore 2. Il sistema ricerca l'identificatore scelto dall'utente nelle tabelle delle annotazioni 3. Lucene restituisce una collezione delle righe relative agli identificatori cercati 4. Il sistema presenta all'utente i risultati della ricerca
Alternative	1. Il sistema riconosce un testo illegale e chiede all'utente di effettuare una ricerca con un nuovo testo 2. Il sistema non trova una corrispondenza nell'indice e notifica all'utente l'assenza di risultati della ricerca
Postcondizioni	Nessuna

4.4.3 Struttura dei Document creati

Come già detto Lucene indicizza un unica tipologia di dato detto Document, istanza di org.apache.lucene.document.Document, e verso questa tipologia di dato bisogna convertire tutti i dati da indicizzare. Quindi il contenuto tabellato nei dati normalizzati, nelle liste di geni regolati e nelle annotazioni, deve essere in qualche modo convertito in istanze della classe Document. Ogni document è costituito da una collezione variabile di Field, campi completamente testuali, a loro volta costituiti da coppie nome, valore.

Quando si effettua, tramite la classe IndexSearcher dell'API, una query, Lucene restituisce al programmatore un oggetto Hits che contiene i riferimenti ai Document trovati. Il recupero effettivo dei documenti per mostrare all'utente i risultati della ricerca è a cura del programmatore e può avvenire recuperando i dati da un DBMS, dal file system, o dallo stesso indice se in fase di indicizzazione si è chiesto a Lucene di archiviare i documenti.

Già in questa fase è facile comprendere l'inutilità di un motore di ricerca che, cercato un gene, restituisca un riferimento a tutti i dati normalizzati e le liste di geni regolati che ne contengono l'identificatore. L'utilizzo di una particolare piattaforma per un esperimento sul DNA di un certo organismo, infatti, già implica l'interessamento dei geni di quell'organismo in quell'esperimento. Allora a che serve la ricerca?

Noi non vogliamo sapere solo in quali esperimenti è stato studiato il livello di espressione di un gene; vogliamo anche estrarre il comportamento del gene dal grosso dei dati per mostrarlo all'utente come risultato della ricerca. Non ci interessa, in sostanza, avere un riferimento ai dati normalizzati di un certo esperimento, ma ci interessa un riferimento ad un punto esatto di tali dati: la riga in cui sono raccolti i valori numerici di espressione del gene.

Da queste considerazioni è scaturita una forte scelta progettuale: creare un Document a partire dal contenuto di ogni riga dei dati e associare ad ogni Document creato un insieme di metadati che consentissero di risalire dalla riga all'intera tabella di dati, alla piattaforma utilizzata e all'esperimento o all'analisi.

Il sistema creerà, quindi, un Document a partire da ogni riga dei dati (dati normalizzati, annotazioni o geni regolati) invece che creare un unico Document per ogni dato. Siamo poi andati incontro ai requisiti approfittando della possibilità di definire per ogni Document da indicizzare un diverso numero di Field, con diversi nomi e con diverse modalità di indicizzazione e archiviazione.

4.4.3.1 I Document per i dati normalizzati

Field	Descrizione	Index	Store
experiment_id	Contiene il valore della chiave primaria della tabella “experiments” del database, corrispondente all'esperimento cui appartengono i dati normalizzati	UN_TOKENIZED	YES
platform_id	Contiene il valore della chiave primaria della tabella “platforms” del database, corrispondente alla array su cui è stato effettuato l'esperimento	UN_TOKENIZED	YES
table	Contiene il nome della tabella del database in cui è fisicamente contenuto il dato. In questo caso vale sempre “experiments”	UN_TOKENIZED	YES
category	Il tipo di dato, in questo caso vale sempre “normalizeddata”	UN_TOKENIZED	YES

Field	Descrizione	Index	Store
row	Il numero di sequenza della riga da cui è stato generato il Document. Vale “0” se è la riga di intestazione	UN_TOKENIZED	YES
targetid	Il targetid contenuto nella riga, se la tabella dei dati normalizzati utilizza tale identificativo. Il Field è vuoto se è la riga di intestazione	UN_TOKENIZED	YES
probeid	Il probeid contenuto nella riga, se la tabella dei dati normalizzati utilizza tale identificativo. Il Field è vuoto se è la riga di intestazione	UN_TOKENIZED	YES
content	Il contenuto dell'intera riga estratta dai dati normalizzati	NO	COMPRESS

4.4.3.2 I Document per le liste di geni regolati

Field	Descrizione	Index	Store
analysis_id	Contiene il valore della chiave primaria della tabella “analyses” del database, corrispondente all'analisi cui appartiene la lista di geni regolati	UN_TOKENIZED	YES
experiment_id	Contiene il valore della chiave primaria della tabella “experiments” del database, corrispondente all'esperimento cui appartiene l'analisi	UN_TOKENIZED	YES
platform_id	Contiene il valore della chiave primaria della ta-	UN_TOKENIZED	YES

Field	Descrizione	Index	Store
	bella platforms, corrispondente alla array su cui è stato effettuato l'esperimento		
table	Contiene il nome della tabella del database in cui è fisicamente contenuto il dato. In questo caso vale sempre “analyses”	UN_TOKENIZED	YES
category	Il tipo di dato, in questo caso vale sempre “regulatedgenes”	UN_TOKENIZED	YES
row	Il numero di sequenza della riga da cui è stato generato il Document. Vale “0” se è la riga di intestazione	UN_TOKENIZED	YES
targetid	Il targetid contenuto nella riga, se lista dei geni regolati utilizza tale identificativo. Il Field è vuoto se è la riga di intestazione	UN_TOKENIZED	YES
probeid	Il probeid contenuto nella riga, se la lista dei geni regolati utilizza tale identificativo. Il Field è vuoto se è la riga di intestazione	UN_TOKENIZED	YES
content	Il contenuto dell'intera riga estratta dalla lista dei geni regolati	NO	COMPRESS

4.4.3.3 I Document per le tabelle delle annotazioni

Field	Descrizione	Index	Store
platform_id	Contiene il valore della	UN_TOKENIZE	YES

Field	Descrizione	Index	Store
	chiave primaria della tabella “platforms” del database, corrispondente alla array cui è associata la tabella delle annotazioni	D	
table	Contiene il nome della tabella del database in cui è fisicamente contenuto il dato. In questo caso vale sempre “platforms”	UN_TOKENIZED	YES
category	Il tipo di dato, in questo caso vale sempre “notestable”	UN_TOKENIZED	YES
row	Il numero di sequenza della riga da cui è stato generato il Document. Vale “0” se è la riga di intestazione	UN_TOKENIZED	YES
targetid	Il targetid contenuto nella riga, se la tabella delle annotazioni utilizza tale identificativo. Il Field è vuoto se è la riga di intestazione	UN_TOKENIZED	YES
probeid	Il probeid contenuto nella riga, se la tabella delle annotazioni utilizza tale identificativo. Il Field è vuoto se è la riga di intestazione	UN_TOKENIZED	YES
content	Il contenuto dell'intera riga estratta dalla tabella delle annotazioni	TOKENIZED	COMPRESS

4.4.4 Scelta delle modalità di indicizzazione e archiviazione

Nelle tabelle riportate nei paragrafi immediatamente precedenti è stato fatto uso di particolari diciture per specificare le modalità di indicizzazione e archiviazione di ogni Field. Aggiungeremo ora qualche dettaglio in più sulle scelte effettuate.

Modalità di indicizzazione:

- **NO**

  Non indicizza il valore del Field, che quindi non può essere cercato.

- **UN_TOKENIZED**

  Indicizza il valore del Field senza processarlo con un Analyzer, in pratica senza filtrare le stop words, senza effettuare lo stemming, senza alterare il case ecc. in modo che questo possa essere cercato così com'è.

- **TOKENIZED**

  Indicizza il valore del Field in modo che possa essere cercato.

Modalità di archiviazione:

- **NO**

  Non immagazzina il valore del Field nell'indice.

- **YES**

  Immagazzina il valore originale del Field nell'indice.

- **COMPRESS**
  Immagazzina il valore originale del Field nell'indice in forma compressa.

I meta dati identificatori di ogni Document sono stati indicizzati in modalità UN_TOKENIZED e archiviati nell'indice. Ciò consente di effettuare la ricerca in tali campi solo ed esclusivamente utilizzandone il contenuto così com'è, oltre che di recuperarli come risultati della ricerca. Tali meta dati consentono di risalire dai Document indicizzati al contenuto delle tabelle del database relazionale; costituiscono il cardine attorno a cui ruota la sincronizzazione dei contenuti dei due sottosistemi.

Il Field “content” dei Document relativi a dati normalizzati e liste di geni regolati è stato archiviato in forma compressa per risparmiare spazio e non è stato indicizzato in quanto il contenuto delle righe estratte dai dati normalizzati e dalle liste di geni regolati è costituito da soli valori numerici, di scarsa rilevanza per la ricerca.

Il Field “content” dei Document relativi alle tabelle delle annotazioni, sempre archiviato in forma compressa, è stato indicizzato e processato da un Analyzer perchè le righe delle tabelle delle annotazioni hanno un contenuto testuale, talvolta espresso in linguaggio naturale, centrale nella ricerca.

Concludendo si noterà come gli unici Field indicizzati sono stati quelli contenenti identificatori e quello contenente le righe delle tabelle delle annotazioni. Lo scopo è quello di effettuare ricerche sulla tabella delle annotazioni, il manifest dell'array, carpire gli identificatori probeid e targetid corrispondenti ai risultati della ricerca per poi effettuare una nuova ricerca, stavolta su dati normalizzati e geni selezionati, con chiave gli identificatori ottenuti al primo passo.

Emerge come l'indicizzazione abbia introdotto una ridondanza nei dati archiviati. Se da un lato era prevedibile che la memorizzazione su disco di un indice richiedesse nuove risorse, l'archiviazione dei Field nell'indice stesso ha comportato una duplicazione di ogni dato archiviato. Dati normalizzati, liste di geni regolati e tabelle delle annotazioni, già memorizzate nei blob testuali del database

si trovano ad essere duplicati nei Document archiviati nell'indice di Lucene. La sostanziale differenza è che i dati sono memorizzati per intero, nel database, divisi in Document contenenti singole righe, nell'indice di Lucene.

Le motivazioni che hanno portato alla duplice archiviazione dei dati sono molteplici, ma riconducibili sempre all'ottenimento di risultati della ricerca di qualità. Volevamo ottenere come risultato della ricerca un insieme di tabelle ognuna costituita da due righe: l'intestazione del dato di partenza e la riga contenente il risultato della ricerca vero e proprio, in modo da conoscere, per ogni valore numerico risultante, il significato.

L'unico modo per ottenere un risultato del tipo presentato, senza effettuare dispendiose scansioni sequenziali dei dati contenuti nel database, stava nell'affidare a Lucene il compito di archiviare i dati, oltre che di indicizzarli. L'API di Lucene, comunque, ci è venuta incontro, consentendoci di archiviare in forma compressa il contenuto delle righe. L'osservazione della dimensione dell'indice creato dai dati utilizzati per il testing ha mostrato come questo, inclusi i documenti archiviati, occupi uno spazio pari al 50% circa della dimensione dei dati originari.

In verità bisogna dire che Lucene, come la maggior parte dei motori di ricerca, è in grado di restituire come risultato della ricerca anche un estratto di poche parole del testo di partenza. Utilizzando un Fragmenter che frammenta il testo, e uno Scorer che associa ad ogni frammento creato un punteggio relativo alla ricerca, Lucene può estrarre dai Document risultato della ricerca la porzione di testo che più si è avvicinata alla query. Tale soluzione, però, non consentiva l'estrazione di una riga precisa, ma solo l'estrazione di un numero n di parole attorno al risultato della ricerca, di fatto non consentendo di tabellare tale risultato. Per questo è stata scartata.

4.4.5 Scelta della modalità di analisi del testo

Prima che un testo sia indicizzato, Lucene lo analizza compiendo una azione detta di tokenizzazione. La tokenizzazione consistente in una trasformazione del testo, estrazione di singole parole (token) utilizzabili nell'indice, e definitiva eliminazione di tutto il resto. In Lucene il compito della tokenizzazione del testo è affidato ad implementazioni della classe astratta Analyzer.

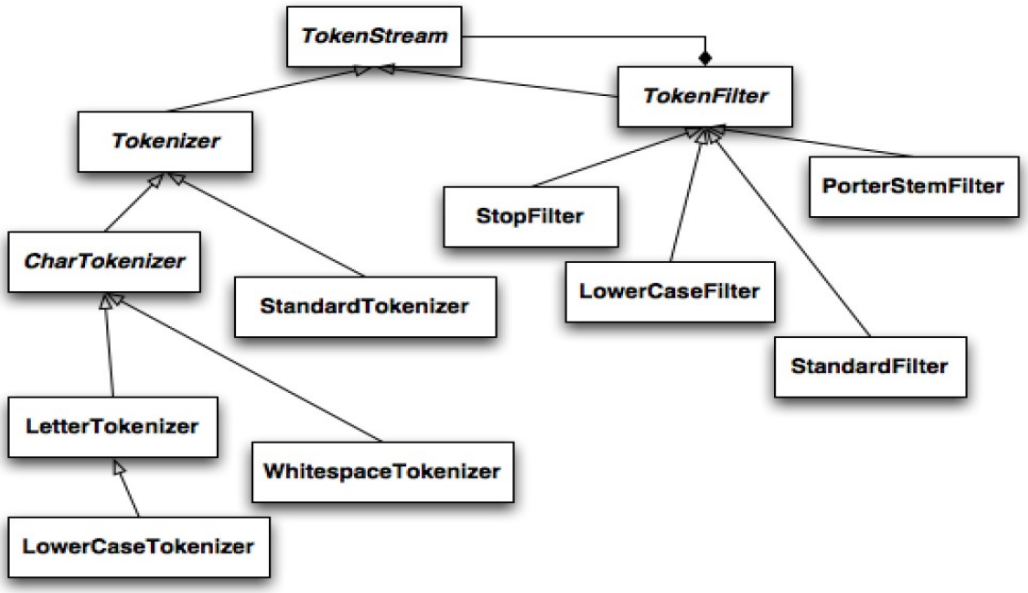

Figura 4.7.: Gerarchia della classe TokenStream, output di ogni Analyzer.

Un Analyzer, quindi, effettua la semplice azione di trasformare un comune testo in un insieme di token. Quello che più conta, però, è la modalità con cui esso genera i token. Gli Analyzer, infatti, hanno una struttura a blocchi; ogni blocco effettua una semplice azione e offre il proprio output al blocco successivo. Una tipica implementazione utilizza un Tokenizer per spezzettare il testo in ingresso in un TokenStream, uno stream di semplici Token, e poi applica ai Token generati uno o più TokenFilter, filtri che li trasformano.

Le azioni più comuni compiute dai filtri sui Token sono le seguenti:

- abbassare il case delle parole individuate (LowerCaseFilter);
- rimuovere le stop words (StopFilter);
- effettuare lo stemming, cioè la sostituzione di ogni parola con la sua radice, ad esempio, alle parole “country” e “countries” viene sostituita la parola “countri” (PorterStemFilter).

La lingua in cui è espresso il testo da analizzare ha ovvie implicazioni sui filtri indicati. Il LowerCaseFilter, ad esempio, potrebbe non aver senso nell'utilizzo con le lingue asiatiche. Liste di stop words nella lingua da analizzare sono necessarie al funzionamento dello StopFilter. La conoscenza della grammatica della lingua in cui è espresso il testo, infine, è necessaria per poter effettuare lo stemming.

Nel nostro caso, ogni eventuale problema di internazionalizzazione dell'indice, comunque, è annullato da due fattori: i manifest (tabelle delle annotazioni) di cui siamo a conoscenza sono esclusivamente in lingua inglese; gli identificativi alfanumerici non sono espressioni di una particolare lingua, se non per l'essere espressi in caratteri occidentali.

A partire dal testo di input, un Analyzer non genererà necessariamente un token per ogni parola, ma genererà un insieme di token il cui contenuto dipende dalle operazioni effettuate. L'abbassamento o meno del case dei token individuati determina la possibilità di effettuare ricerche case sensitive o insensitive. La rimozione delle stop words velocizza le ricerche e snellisce l'indice. Lo stemming consente di estendere la sensibilità della ricerca, ad esempio, a tutte le possibili coniugazioni di un verbo a partire da una sola di esse, indifferentemente al maschile o al femminile di un agettivo, al singolare o al plurale di un nome.

Lo sviluppatore ha così la possibilità di creare una propria implementazione della classe Analyzer, customizzata sulle esigenze della

ricerca da effettuare. Tuttavia Lucene fornisce allo sviluppatore una serie di Analyzer di uso comune, alcuni particolarmente semplici come il SimpleAnalyzer, altri molto elaborati come lo StandardAnalyzer. Riportiamo in tabella una panoramica degli Analyzer di più frequente utilizzo.

Analyzer	Passi intrapresi
WhitespaceAnalyzer	Costruisce i token dividendo il testo ad ogni whitespace.
SimpleAnalyzer	Costruisce i token dividendo il testo ad ogni carattere non letterale, poi abbassa il case
StopAnalyzer	Costruisce i token dividendo il testo ad ogni carattere non letterale, abbassa il case e rimuove le stop words
KeywordAnalyzer	Trasforma l'intero testo in un singolo Token e non applica alcun filtro. E' utile per l'indicizzazione di dati come i codici postali, gli identificatori, i nomi di prodotti e tutti i codici in generale.
StandardAnalyzer	Costruisce i token utilizzando un sofisticato tokenizer, lo StandardTokenizer, in grado di riconoscere e-mail, indirizzi, acronimi, nomi propri, stringhe alfanumeriche e molto altro ancora; abbassa il case e rimuove le stop words

L'analisi dei dati da indicizzare e i test effettuati hanno indicato come migliore soluzione l'utilizzo dello StandardAnalyzer. Tra tutti i Document creati, solo il Field “content” associato alle tabelle delle annotazioni viene indicizzato in modalità TOKENIZED, cioè utilizzando un Analyzer. Questo campo contiene una intera riga della tabella delle annotazioni, quindi contiene identificatori alfanumerici come il probeid e il targetid, il nome del gene e i sinonimi, alfanumerici anche questi, e del testo in linguaggio naturale, ad esempio nelle colonne riguardanti l'ontologia.

Lo StandardAnalyzer si è dimostrato in grado di indicizzare ogni identificativo, sigla, nome di gene, alfanumerico o comune testo, oltre che le parole in linguaggio naturale, in accordo a quanto indi-

cato nella documentazione di Lucene. Per questo motivo è stata scartata l'ipotesi di costruire un Analyzer ad hoc per il nostro sistema.

4.4.6 I passi della ricerca

Una volta popolato l'indice, strutturato come si è visto nei paragrafi immediatamente precedenti, è possibile utilizzare l'API di Lucene per effettuare delle ricerche. Anche in questo caso l'interfaccia offerta al programmatore è particolarmente semplice: l'oggetto IndexSearcher si occupa di effettuare una Query sull'indice. Le query possono essere costruite dal programmatore e/o generate da un QueryParser a partire dalla stringa di testo immessa dall'utente.

E' stata progettata e sviluppata una procedura di ricerca diversa per ognuno dei casi d'uso individuati. Gli elementi chiave per comprendere il funzionamento delle diverse procedure di ricerca sono elencati in seguito come afferamazioni:

- un probeid identifica univocamente i probe dell'array costituendo il punto in corrispondenza del quale si effettuano le misurazioni;
- più probeid possono essere raggruppati sotto il nome di uno stesso targetid;
- un gene può essere collegato ad uno o più probe;
- il livello di espressione di un gene è noto a partire dalle misure effettuate in corrispondenza del probe, o dei probe, cui esso è collegato;
- cercare un gene vuol dire cercare il probe, o il gruppo di probe cui esso è collegato sull'array.

Di conseguenza un ricercatore, volendo cercare il comportamento di un gene, dovrebbe andare alla ricerca di un probeid o di un targetid. Abbiamo ritenuto poco maneggevole l'utilizzo dell'identificatore di un probe o di un target nella ricerca di un gene, per questo abbiamo deciso di automatizzare la procedura di ricerca,

sfruttando l'associazione probeid/targetid/gene codificata in ogni tabella delle annotazioni.

A seguire è riportata una descrizione dei passi compiuti per realizzare il più complesso dei casi d'uso per la ricerca: la ricerca in tutti i dati attraverso le tabelle delle annotazioni. Gli altri casi d'uso si risolvono sempre nel compiere un sottoinsieme delle operazioni compiute in questo caso, ad esempio nel compiere solo il primo passo nel caso in cui si cerchino i dettagli su un identificatore nelle tabelle delle annotazioni.

La ricerca parte dall'immissione di una stringa di testo nell'apposito form della pagina di ricerca. A partire dalla stringa immessa dall'utente il sistema genera, per mezzo dell'oggetto QueryParser, una istanza dell'oggetto Query. La query così generata viene effettuata nel solo campo “content” dei Document per i quali il campo “category” assume il valore “notestable”, vale a dire solo sulle righe delle tabelle delle annotazioni.

In risposta a questa prima query, Lucene restituirà un oggetto Hits, contenente i riferimenti ai documenti risultato della ricerca. Di ognuno di questi Document sarà estratto il contenuto del campo “targetid” e il contenuto del campo “probeid”. In questo modo il sistema sarà a conoscenza di tutti gli identificatori associati al gene o ai geni risultato della ricerca.

Una nuova query viene costruita dal sistema, sotto forma di query booleana, a partire dagli identificatori raccolti. Stavolta sarà effettuata la ricerca degli identificatori raccolti nei campi “probeid” e “targetid” di tutti i Document caratterizzati da un campo “category” contenente il valore “normalizeddata” o il valore “regulatedgenes”. La query andrà cioè a cercare gli identificatori tra i dati normalizzati e le liste di geni regolati.

In risposta a questa seconda query Lucene restituirà un oggetto Hits contenente i riferimenti ai Document risultato della ricerca. Il sistema a questo punto provvederà a raccogliere il campo “content” di questi document, cioè il conenuto di ogni riga e a tabellarlo congiuntamente all'intestazione della tabella, a sua volta recuperata effettuando una query sull'indice.

In figura 4.7 è riportato un activity diagram relativo all'attività di ricerca. L'ultimo blocco prima della visualizzazione da parte dell'utente rappresenta il filtraggio con cui il sistema, dopo la preparazione dei risultati della ricerca, nasconde tutti i dati degli esperimenti che non rientrano nell'ambito di visibilità dell'utente stesso.

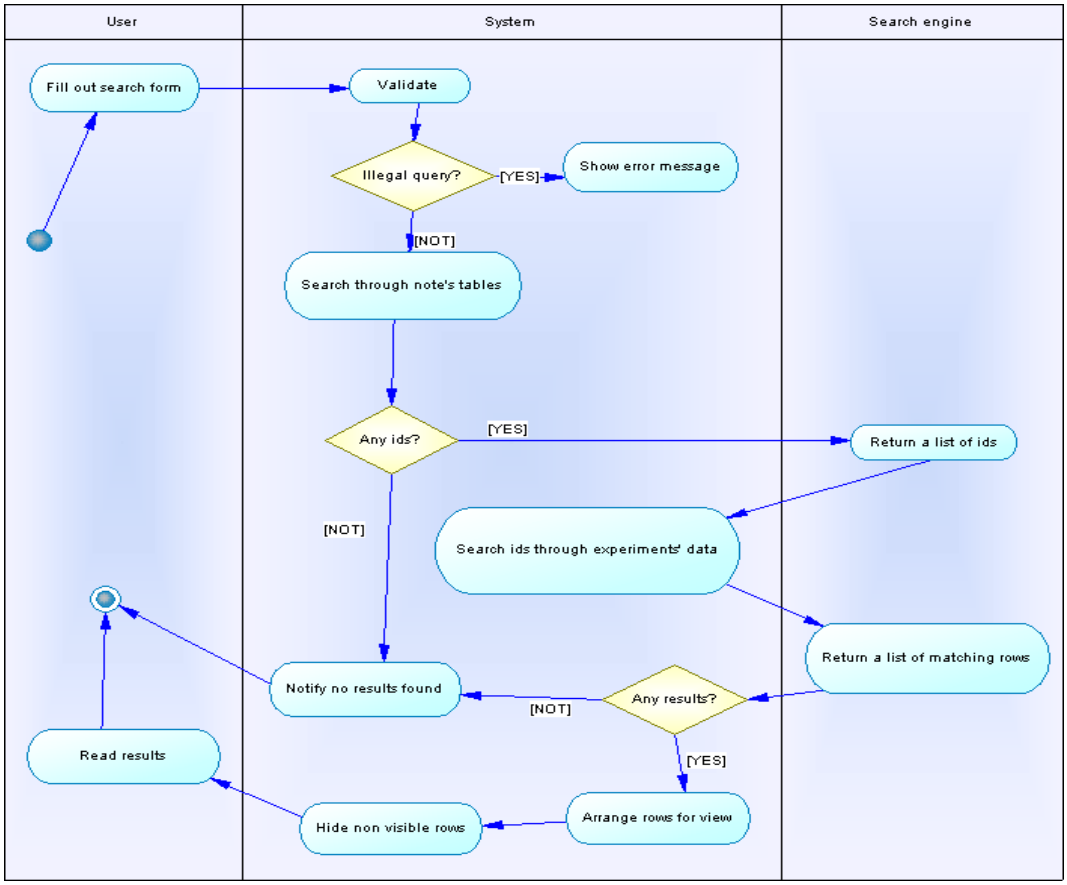

Figura 4.8.: Activity diagram della procedura di ricerca automatizzata.

4.4.7 Indicizzazione incrementale e batch

Come già preannunciato, Apache Lucene offre strumenti per la gestione di una indicizzazione incrementale o di una indicizzazione batch. L'indicizzazione incrementale consiste nella possibilità di indicizzare un Document ogni qual volta un nuovo documento venga inserito nel sistema. L'indicizzazione batch consiste, naturalmente, nell'indicizzazione di lotti di documenti, una tantum, ad intervalli periodici, di solito nelle ore di minore carico del sistema.

Lucene, lavorando su copie diverse dell'indice, provvede a gestire contemporaneamente i processi di indicizzazione e ricerca. In questo modo, mentre un IndexWriter sta effettuando l'indicizzazione di un nuovo Document, un IndexReader, che lavora su una diversa copia dell'indice, consente all'utente di effettuare ricerche tra i Document precedentemente indicizzati. Il tutto è gestito in modo completamente trasparente, sgravando lo sviluppatore di ogni responsabilità.

L'indicizzazione di un singolo Document in generale non è un processo che richieda un grosso impegno della CPU, della memoria e del disco rigido. Tuttavia, la mole dei dati da indicizzare nel nostro caso, decine di migliaia di Document derivanti da file di testo che raggiungono facilmente i 20Mbyte, ha imposto una riflessione ulteriore sull'utilizzo delle risorse hardware e sull'impatto che i tempi di attesa hanno sull'utilizzo del sistema.

Dai test effettuati facendo girare l'applicazione su medeaserver.isa.cnr.it, macchina equipaggiata con un processore Intel Pentium 4 a 3GHz, due Gbyte di memoria fisica DDR2 a 533MHz e una coppia di Dichi RAID 1 da 7200rpm, è risultata una durata del processo di indicizzazione e archiviazione nel database di circa 40/50 secondi per 10 Mbyte di dati. Riferirsi ai tempi di elaborazione in modo non troppo preciso è una necessità. La durata esatta del processo di

indicizzazione, infatti, è condizionata, oltre che dalla mole di dati da indicizzare, dai seguenti fattori:

- la connettività. Il trasferimento di qualche decina di Mbyte di dati a partire da una macchina appartenente alla stessa LAN cui appartiene il server è quasi istantaneo; non si può dire altrettanto quando i dati provengono da un punto qualsiasi del Web;
- i dischi rigidi. Il processo di indicizzazione si risolve, prima o poi, con una serie di scritture su disco per la memorizzazione dell'indice stesso: prestazioni diverse corrispondono a dischi da 5400 rpm, da 7200 rpm, da 10000 rpm e così via. Discorso analogo per quanto riguarda la banda passante verso i dischi: cofigurazioni RAID diverse dal semplice mirroring avrebbero dato migliori prestazioni;
- CPU e memoria fisica. Pur non trattandosi di un processo number crunching, abbiamo rilevato un consistente impegno della CPU durante il processo di indicizzazione. Inoltre la quantità di memoria fisica realmente disponibile, trattandosi di una applicazione Java, dipende dai paramentri di configurazione della Java Virtual Machine;
- il carico di lavoro cui è sottoposto il server. Le prestazioni di un web server su cui girano più applicazioni sono necessariamente legate al numero di request servite;
- I parametri di tuning del processo di indicizzazione. Lucene consente, agendo su un insieme di parametri che vedremo in seguito, la sintonizzazione del processo di indicizzazione alle performance della macchina su cui gira il processo.

Le difficoltà nello stimare l'incidenza del contributo di ognuno dei fattori indicati sulla durata del processo di indicizzazione è evidente. Tuttavia, facendo girare l'applicazione su più di una macchina, ci siamo fatti un idea, seppur vaga, della durata del processo e, a partire da questa, abbiamo effettuato le nostre considerazioni riguardo all'impatto dei tempi di attesa sull'utilizzo dell'applicazione.

E' apparso subito evidente come il lancio del processo di indicizzazione congiuntamente all'inserimento dei dati nel database potesse comportare un attesa troppo lunga per l'utente. Per questo motivo abbiamo deciso di lasciare all'utente stesso la scelta di quando indicizzare i dati: subito, all'atto dell'inserimento, o in un processo batch, schedulato automaticamente dal sistema nelle ore di minor carico.

La prima soluzione, l'indicizzazione dei dati all'inserimento, comporta l'indubbio vantaggio di rendere i dati subito disponibili per la ricerca. Tuttavia l'utente dovrà mettere in conto un tempo di attesa maggiore, prima che gli sia notificato l'avvenuto inserimento, e ciò comporterà un fastidio quando l'utente tenti l'inserimento di più di un esperimento in maniera consecutiva. Inoltre, l'indicizzazione immediata avverrà quasi certamente nel corso dell'orario di lavoro del laboratorio, cioè nell'orario di maggior carico per la macchina servente.

La seconda soluzione, l'indicizzazione batch, accorcia i tempi di attesa dell'utente rendendo di fatto più piacevole l'utilizzo del sistema; ritarda l'indicizzazione dei dati inseriti ad un orario di scarso carico del server rendendo meno probabile il sovraccarico dello stesso. Lo svantaggio di questa soluzione è essenzialmente nel dover attendere il processo batch prima che si possano effettuare ricerche sui nuovi dati inseriti.

Da quanto detto sembrerebbe utile orientarsi verso l'implementazione della sola soluzione batch. La soluzione incrementale, infatti, potrebbe addirittura consentire il sabotaggio del sistema da parte di utenti malintenzionati. Tuttavia, immaginando l'utilizzo dell'applicazione all'interno della rete di un istituto di ricerca, con la possibilità di uno stretto controllo sull'attività degli utenti, abbiamo deciso di implementare entrambe le soluzioni.

Si è deciso di configurare lo scheduler che si occupa dell'indicizzazione batch per l'esecuzione alle 2:00 di ogni notte. In questo

modo, i dati che non sono stati indicizzati subito saranno disponibili per la ricerca al massimo entro 24 ore. Inoltre il processo di indicizzazione batch scriverà un log delle operazioni effettuate e lo invierà con una email all'amministratore del sistema notificando il corretto funzionamento o meno del processo e i problemi incontrati. Inoltre, nel caso in cui si presentino problemi di indicizzazione, il sistema provvederà all'invio di una ulteriore e-mail all'utente che ha immesso i dati non indicizzabili, chiedendo di contattare al più presto l'amministratore.

4.4.8 Sincronizzazione di database e indice

La ridondanza introdotta nei dati memorizzati, consistente nella duplice archiviazione, per righe sotto forma di Document nell'indice e per esteso nel database, e la costruzione stessa di un indice slegato dal contenuto del database, hanno imposto alla nostra attenzione il problema dell'allineamento del contenuto dell'indice al contenuto del database.

Bisognava garantire, in sostanza, che le ricerche dell'utente fossero sempre eseguite su esperimenti effettivamente memorizzati nel database; che alla rimozione di un esperimento dal database corrispondesse la rimozione dei suoi Document dall'indice; che all'aggiornamento di ognuno dei dati memorizzati nel database corrispondesse l'aggiornamento dei dati indicizzati.

La risposta a tale problema doveva necessariamente essere formulata a livello applicativo. Per questo sono stati previsti una serie di automatismi che ruotano attorno alle procedure di inserimento, modifica e eliminazione dei dati nel sistema. Un flag, una variabile booleana archiviata nel database, notifica l'avvenuta indicizzazione dei dati.

Le tabelle “platforms”, “experiments” e “analyses” contenenti ognuna, nella forma di un blob testuale, rispettivamente una tabella delle annotazioni, un file di dati normalizzati e uno di geni regolati, sono state tutte dotate di una colonna “indexed”. Il contenuto della colonna indexed viene settato a true al termine dell'indicizzazione del dato interessato; assume valore false, invece, qualora il dato non sia stato ancora indicizzato.

Quando l'utente sceglie l'indicizzazione immediata, il setting dell'attributo indexed, che di default assume valore false, viene effettuato in una query distinta dalla query di inserimento dei dati nelle tabelle. In quest modo, sfruttando il supporto transazionale, offerto da MySQL, e inserendo il processo di indicizzazione a metà strada

tra l'inserimento dell'esperimento, analisi o piattaforma nel database e il setting dell'attributo indexed, ci siamo tutelati rispetto a eventuali eccezioni sollevate dal processo di indicizzazione stesso. Nel caso in cui una eccezione sia sollevata dal processo di indicizzazione dei dati, infatti, l'utilizzo di una transazione ci garantisce che anche l'inserimento nel database non giunga al commit, cioè non diventi persistente.

Quando l'utente sceglie di affidare i propri dati al processo di indicizzazione batch, i dati vengono archiviati nel database con valore false nel campo indexed. Dalla ricerca dei dati per i quali il campo indexed è settato a false, il processo di indicizzazione batch individuerà i dati non ancora indicizzati. Esso preleverà uno per uno i dati per i quali indexed è false, ne leggerà i blob, lancerà l'indicizzazione e al termine setterà il campo indexed corrispondente al valore true.

Abbiamo scelto di estendere la possibilità di scegliere l'indicizzazione immediata o batch anche alle attività di aggiornamento dei dati. All'aggiornamento dei dati presenti nel database corrisponderà la rimozione dall'indice dei vecchi dati e il setting dell'attributo indexed a false, se l'utente ha scelto l'indicizzazione batch; la rimozione dall'indice dei vecchi dati, l'indicizzazione dei nuovi e il setting del campo indexed a true, se l'utente ha scelto l'indicizzazione immediata.

Per quanto riguarda l'eliminazione dei dati, l'allineamento dell'indice al database è più semplice: ad ogni rimozione dal database, segue la rimozione dall'indice.

5 Implementazione dell'applicazione

Contenuto del capitolo

5.1 Introduzione

Dopo aver ampiamente illustrato cosa intendevamo realizzare e con quale piattaforma tecnologica volevamo realizzarlo, ora daremo qualche dettaglio in più sul come abbiamo realizzato l'applicazione. Avendo fatto uso di molte delle cosiddette best practices, per quanto attiene lo sviluppo di applicazioni web con Struts, non illustreremo in dettaglio tutti gli aspetti del software sviluppato ma ci soffermeremo su quegli aspetti che, a nostro parere, fanno dell'applicazione realizzata un prodotto di elevato livello qualitativo.

Il processo di sviluppo centrato su Apache Struts è ormai standardizzato, consolidato e, soprattutto, largamente documentato. Per questo motivo rimandiamo ai testi dedicati al celebre framework, di cui una selezione è disponibile in bibliografia, chi voglia approfondire gli aspetti realizzativi più comuni. Illustreremo invece in dettaglio le tecniche utilizzate per realizzare le funzionalità peculiari del business model e per risolvere i problemi scaturiti dall'adattamento delle tecnologie utilizzate al modello stesso.

5.2 DataSource e connection pooling

Un DataSource è la rappresentazione di una sorgente dati nel linguaggio di programmazione Java. Le applicazioni accedono alle sorgenti di dati tramite connessioni, quindi una istanza di DataSource può essere pensata come una factory di connessioni verso una particolare sorgente. Nel nostro caso esso è semplicemente l'interfaccia da utilizzare per l'accesso al driver JDBC, MySQL Connector Java.

Un connection pool è una cache delle connessioni a un database mantenuta in modo da riutilizzare le connessioni create per servire future richieste. I connection pool sono utilizzati per migliorare le performance dei comandi eseguiti su un database: aprire e tenere aperta una connessione per ogni utente, specialmente quando le richieste provengono da una applicazione web è costoso e porta ad uno spreco di risorse.

Nel connection pooling, dopo che una connessione è stata creata, questa viene inserita nel pool di connessioni e viene riutilizzata quando si verifica una nuova richiesta di servizio, in modo che non sia necessario stabilire una nuova connessione ogni volta. Quando tutte le connessioni del pool sono occupate, una nuova connessione viene creata ed aggiunta al pool.

La tecnica illustrata ha il vantaggio di tagliare i tempi di attesa dell'utente per stabilire una connessione col database. Creare una nuova connessione, infatti, spesso impiega un tempo molto maggiore del tempo necessario a portare a termine una transazione.

Tra i molti database connection pool disponibili nel mondo open source, abbiamo scelto di utilizzare quello proposto da Apache: il package commons-dbcp, per altro consigliato dagli stessi sviluppatori di Struts. Le applicazioni possono utilizzare DBCP direttamente o attraverso una interfaccia esistente del servlet engine o del framework. Abbiamo scelto di utilizzare il DataSource manager di Struts, configurando il connection pool nel file di configurazione struts-config.xml come segue.

```
<data-sources>
      <data-source key="microarray2"
      type="org.apache.commons.dbcp.BasicDataSource">
            <set-property property="url"
      value="jdbc:mysql://medeaserver.isa.cnr.it:
      3306/microarray2"/>
      <set-property property="password" value="password"/>
      <set-property property="driverClassName"
      value="com.mysql.jdbc.Driver"/>
      <set-property property="description" value="Microarray
      Data Base"/>
      <set-property property="username" value="username"/>
      </data-source>
</data-sources>
```

In particolare è stato configurato il DataSource BasicDataSource, una implementazione dell'interfaccia javax.sql.DataSource che utilizza la tecnica del pooling.

5.3 Supporto transazionale

Una transazione è una unità logica di elaborazione richiesta da una applicazione che da luogo ad una serie di operazioni fisiche elementari (una serie di letture e scritture) su una base di dati. Utilizzare il supporto transazionale consente di eseguire più comandi, statement nel linguaggio SQL, come se fossero uno solo, in modo che, o sono eseguiti tutti, o nessuno.

Una transazione deve essere dotata delle cosiddette proprietà ACID, Atomicità, Consistenza, Isolamento e Durability. Quindi essa è una unità elaborativa, nel senso che non è possibile eseguirla solo in parte; deve rispettare i vincoli di integrità imposti sui dati; deve portare a risultati indipendenti da quelli di ogni altra operazione in corso sulla base di dati; deve garantire la persistenza delle modifiche apportate.

Nel modello transazionale una transazione può terminare correttamente o non correttamente. Una transazione termina correttamente quando l'utente, o una applicazione per mezzo del driver JDBC, esegue l'istruzione di COMMIT o COMMIT WORK; termina non correttamente in due casi distinti: o perché l'utente, o ancora l'applicazione, ha deciso che non ha senso continuare ed ha effettuato un ROLLBACK, oppure perché un guasto tecnico o la violazione di un vincolo ne ha causato l'interruzione (ABORT).

In JDBC le transazioni sono controllate tramite l'oggetto java.sql.-Connection. Per default, l'oggetto Connection lavora in modalità auto-commit. Ciò vuol dire che una operazione di commit viene eseguita automaticamente al termine dell'esecuzione di ogni statement SQL. In altre parole ogni comando viene eseguito come se fosse una transazione a sé stante.

Tuttavia è possibile disabilitare la modalità auto-commit per far in modo che il commit o il rollback delle transazioni avvenga in modo esplicito, tramite l'invocazione degli appositi metodi. Così ogni volta che abbiamo avuto bisogno di proteggere con una transazione l'esecuzione di più comandi SQL abbiamo usato la sintassi:

```
Connection con = dataSource.getConnection();
con.setAutoCommit(false);
```

abbiamo eseguito i nostri statement, e abbiamo concluso con un

```
con.commit();
```

per concludere la transazione correttamente, o con un

```
con.rollback();
```

per annullare ogni modifica apportata e riportare il database allo stato in cui era prima dell'esecuzione della transazione stessa.

Abbiamo fatto uso del supporto transazionale in molteplici casi, primo fra tutti, l'archiviazione degli esperimenti. Inserire un esperimento nel database, infatti, richiede l'esecuzione di operazioni di INSERT su più tabelle, experiments, analyses, biosamples, experimentalfactorsvalues, statisticalparametersvalues, e in numero variabile, rispettivamente a seconda del numero di analisi, di campioni, di fattori sperimentali e di parametri statistici.

Le tabelle oggetto dell'inserimento, inoltre, sono tra loro legate da vincoli di chiave straniera che impongono che gli inserimenti avvengano in una precisa sequenza. Così, per evitare la possibilità di ritrovarci con un database “sporco”, ad esempio, a causa dell'insorgere di eccezioni durante l'esecuzione degli statement o della violazione di un vincolo, abbiamo raccolto tutti gli INSERT da compiere in un unica unità elaborativa, una transazione appunto.

Discorso analogo è stato fatto ogni qual volta portare a termine una operazione abbia richiesto l'esecuzione di più di uno statement, come nel caso dei metodi di rimozione di esperimenti, analisi e campioni. Altre volte abbiamo creato transazioni ad arte, come nel caso del processo di indicizzazione, per garantire che il contenuto del database fosse sempre allineato al contenuto dell'indice.

5.4 Il logging

La parola logging indica in generale l'attività attraverso la quale un software notifica degli eventi scrivendo messaggi sulla console di sistema, in un file sul disco, in una e-mail e così via. Gli eventi che possono essere oggetto del logging sono limitati solo dalla fantasia dello sviluppatore; in generale si tratta di eventi legati alla sicurezza dell'applicazione, come il log in e log out degli utenti, o malfunzionamenti, come nel caso delle eccezioni.

Molto spesso si pensa al logging come ad una tecnica di debug di basso profilo tecnologico, il che può anche essere vero, ma utilizzando sistemi di logging più sofisticati è anche possibile:

- conoscere dettagli sullo stato di una applicazione al runtime;
- incrementare o decrementare la quantità di dettagli forniti;
- modificare il formato dei messaggi;
- inviare i messaggi ad una destinazione diversa;
- creare molteplici log.

Il framwork utilizzato, Apache Struts, include il package Commons Logging, uno dei tanti progetti Jakarta, tra le proprie librerie. Commons logging è un framework leggerissimo che fornisce una interfaccia comune per effettuare il logging attraverso i package più utilizzati oggi, come Log4J o java.util.logging di Java2SE 1.5. Questa interfaccia comuene consente allo sviluppatore di scegliere il package che si vuole utilizzare senza dover apportare mofiche al codice.

Per utilizzare Commons Logging, occorre innanzi tutto accertarsi di averne importato il file jar nella cartella WEB-INF/lib della propria applicazione. Successivamente, per abilitare i nostri sorgenti al logging, basta importare le due classi seguenti:

```
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
```

Log è la classe che effettivamente si occupa del logging, mentre la classe LogFactory si occupa di ottenere una istanza della classe precedente.

Dopo aver importato le due classi non resta altro che ottenere una istanza di Log da LogFactory passando a quest'utlima il nome della classe che intende invocare il logger.

```
Log log = LogFactory.getLog(Login.class);
```

e stampare i nostri messaggi utilizzando una sintassi del tipo:

```
log.debug("This is my debug message.");
log.info("This is my info message.");
log.warn("This is my warn message.");
log.error("This is my error message.");
log.fatal("This is my fatal message.");
```

L'effettiva stampa dei messaggi avviene, nel nostro caso, utilizzando Log4J. La configurazione di Commons Logging per l'utilizzo di Log4J è automatica, essendo questa la scelta predefinita, basta inserire la libreria nella nostra directory WEB-INF/lib.

Log4J ha bisogno di essere configurato utilizzando un properties file, log4j.properties, da creare nella cartella WEB-INF/classes. A seguire riportiamo la configurazione adottata per la nostra applicazione.

```
log4j.rootCategory=INFO, stdout, R

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.conversionPattern %d{DATE} %5p
      [%t] (%F\:%L) -%m%n

log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=${catalina.base}/logs/wp_log_file.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd'.txt'
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.conversionPattern=%d{DATE} %5p [%t]
      (%F\:%L) - %m%n
```

Abbiamo definito due appender, cioè due destinazioni di output, stdout e R. Il primo, un ConsoleAppender, si occupa della stampa dei messaggi sulla console del sistema. Il secondo, di tipo DailyRollingFileAppender, scrive i messaggi ogni giorno su un nuovo file di testo di cui abbiamo indicato il formato del nome.

5.5 Gestione delle eccezioni

Struts offre la possibilità definire la gestione delle eccezioni in maniera dichiarativa, utilizzando i tag <global-exception> e <exception> nel file di configurazione struts-config.xml. Il metodo execute() di ogni Action class può sollevare una eccezione di tipo Exception. Così facendo si lascia che l'eccezione sia gestita dal RequestProcessor che dapprima determina se l'eccezione è stata dichiarata nel file di configurazione e, se è così, chiama la classe che deve gestire l'eccezione, per default org.apache.struts.action.ExceptionHandler.

L'ExceptionHandler è responsabile della preparazione delle informazioni da inviare al presentation layer. In particolare il gestore crea o raccoglie un ActionError e lo inserisce in un oggetto ActionErrors. Successivamente l'ExceptionHandler determina verso chi indirizzare il flusso di controllo utilizzando un ActionForward.

Nel file di configurazione di Struts è possibile definire eccezioni globali, per l'intera applicazione, o locali, per una singola Action. Le eccezioni così definite, dal punto di vista operativo, sono simili agli ActionForward. Se definiamo un tipo di eccezione come globale, qualunque Action Class la sollevi, questa sarà gestita nello stesso modo, cioè nel modo dichiarato nel file di configurazione. Se invece definiamo una eccezione come locale, allora questa potrà essere gestita in modo diverso dipendentemente dalla Action class che la definisce.

Qualora non sia sufficiente quanto ci viene offerto dall'ExceptionHandler di Struts possiamo crearne uno custom mediante una semplice estensione. Effettuando l'overriding del metodo execute() dell'ExceptionHandler è possibile introdurre nuove e interessanti funzionalità come:

- il logging delle informazioni associate all'eccezione prima che questa sia inviata al presentation layer;
- la gestione di catene di eccezioni;
- l'invio di messaggi e-mail agli amministratori del sistema.

Per definire una eccezione globale si usa una sintassi del tipo seguente:

```
<exception bundle="ApplicationResources"
      handler="microarray.utils.CustomExceptionHandler"
      key="error.RuntimeException" path="page.baderror"
type="java.lang.RuntimeException"/>
```

Così facendo tutte le eccezioni del tipo java.lang.RuntimeException saranno gestite dal nostro CustomExceptionHandler; ad esse sarà associato un ActionError creato a partire dal messaggio error.RuntimeException presente nel resource bundle principale. La gestione dell'eccezione si concluderà, infine, con un forward verso la pagina page.baderror.

Per definire la stessa eccezione come locale, invece, si usa la sintassi XML:

```
<action input="page.login" name="loginForm"
parameter="method"
      path="/login" scope="request"
      type="microarray.actions.LoginAction" validate="false">
      <set-property property="cancellable" value="true"/>
      <forward name="success" path="page.loggedin"/>
      <forward name="failure" path="page.login"/>
      <forward name="login" path="page.login"/>
      <exception key=”login.error”
            type=”java.lang.Exception” />
</action>
```

Se una eccezione di tipo Exception viene sollevata dalla LoginAction, allora l'ExceptionHandler di Struts, provvederà alla creazione un ActionError usando la chiave login.error.

Tra i punti di forza di tale tecnica è evidente la possibilità di deffinire diversi handler, diversi path, diversi messaggi e diversi bundle, per la stessa eccezione quando questa sia sollevata da diverse Action class.

Nello sviluppare l'applicazione abbiamo creato un nostro proprio gestore delle eccezioni, CustomExceptionHandler, e utilizzato largamente le global exceptions di Struts per fare in modo che la nostra applicazione reagisse alle eccezioni mantenendo fermi i seguenti punti:

- l'annotazione nel log e la stampa a console di tutte le informazioni utili al debug;
- la notifica degli errori più gravi all'amministratore tramite posta elettronica;
- la presentazione all'utente di pagine di errore amichevoli e diverse a seconda della gravità del problema.

Inoltre abbiamo ritenuto utile, ai fini della sicurezza, nascondere all'utente ogni ulteriore dettaglio sull'errore che potesse dar modo a questo di risalire alle elaborazioni compiute dall'applicazione. Così, ad esempio, in corrispondenza di una SQLException, l'utente vedrà comparire una pagina che gli notifica un errore nel database, ma non avrà dettagli sulla query che ha generato l'errore o sullo stack di attivazione dei metodi.

5.6 Authentication e Authorization

L'autenticazione, in inglese authentication, è il processo mediante il quale si prova a verificare l'identità dell'utente di un sistema informativo. Nel campo delle applicazioni web questo processo è di norma associato all'inserimento di uno username e di una password. E' un comune errore pensare che il problema dell'autorizzazione, in inglese authorization, coincida con quello dell'autenticazione. Una terminologia più precisa dovrebbe indicare l'autenticazione come il processo in cui si verifica l'identità di una persona, mentre l'autorizzazione è il processo con cui si verifica che una persona a noi nota abbia l'autorità per effettuare una certa operazione. Per questo l'autenticazione deve sempre precedere l'autorizzazione.

Dopo che uno sportello bancomat ci ha “identificato”, riconoscendo valida la coppia costituita dalla carta magnetica inserita e il codice PIN digitato, questo ci consente di accedere alle informazioni sul nostro conto corrente, ma non ci “autorizza” ad accedere a conti che non siano di nostra proprietà. Poiché l'autorizzazione non può avvenire senza che prima sia avvenuta l'autenticazione, quest'ultimo termine è spesso utilizzato per indicare la combinazione di autorizzazione e autenticazione.

La nostra applicazione riserva l'accesso alle proprie funzionalità ai soli utenti registrati che abbiano superato la procedura di autenticazione fornendo una password valida. Invece, gli utenti che non sono in grado di fornire credenziali valide, hanno accesso solo all'area pubblica del portale. Molte applicazioni web restringono l'area pubblica alla sola pagina di log in; noi abbiamo preferito estendere l'area pubblica alla home page, alla pagina per il recupero dei dati personali e alla pagina per la registrazione.

Le specifiche J2EE definiscono un approccio dichiarativo al problema della sicurezza, così il web container utilizzato, Apache Tomcat, integra un proprio sistema di autenticazione e autorizzazione. Noi, ritenendo poco flessibile la soluzione offerta dal servlet engine, abbiamo preferito farci carico dell'implementazione di un sistema di sicurezza interno all'applicazione.

5.6.1 Log in

L'autenticazione, all'interno della nostra applicazione, avviene tramite una comune pagina di log in. L'utente inserisce nome utente e password e, se questi sono validi, un istanza dell'oggetto UserDTO, popolata con i dati prelevati dalla tabella “users” del database, viene aggiunto alla sessione.

```
UserDTO user = lb.validateUser(loginForm.getUsername(),
      loginForm.getPassword());
HttpSession session = request.getSession();
session.setAttribute("user",user);
```

Nel caso in cui le credenziali inserite non siano valide, il sistema ripropone al visitatore la pagina contenente il form da compilare.Alla procedura standard abbiamo aggiunto qualche complicazione. Innanzi tutto, per intralciare i malintenzionati, dopo tre tentativi falliti, invece che mostrare nuovamente la pagina di log in, il sistema mostra una pagina di errore. Inoltre, per velocizzare l'accesso, abbiamo ritenuto utile salvare le credenziali corrette mediante cookies sul pc dell'utente, ovviamente criptandoli per impedirne la lettura.

5.6.1.1 I Cookie

I cookie, in italiano si chiamerebbero biscottini, sono piccoli file di testo che i siti web utilizzano per immagazzinare nel computer dell'utente informazioni che si vuole far sopravvivere alla durata di una sessione. I cookie viaggiano sulla rete sotto forma di header

aggiuntivi in una request o response HTTP: quando il server voglia assegnare un cookie all'utente, lo aggiungerà tra gli header della response. È possibile impostare più cookie in una sola response HTTP.

Un client che riceve dei cookie memorizzerà ognuno di essi sotto forma di file di testo in una specifica directory del proprio file system. Il file di testo generato sarà composto da una stringa di testo ASCII arbitraria, consistente nel contenuto del cookie così come è stato inviato dal server, una data di scadenza e un pattern per riconoscere i domini a cui rimandarlo. Il client rimanderà il cookie, senza alcuna modifica, allegandolo a tutte le request HTTP inviate verso i domini che soddisfano il pattern, entro la data di scadenza.

Solitamente si codificano nei cookie informazioni come la data dell'ultima visita, lo username di un utente, la sua password, gli oggetti aggiunti al carrello della spesa, le impostazioni utilizzate per la ricerca, le preferenze sull'aspetto del sito o qualunque altra informazione che possa essere utile conservare oltre lo scadere della sessione.

Nell'applicazione oggetto di questa tesi abbiamo creato dei cookie per salvare sulla macchina client due informazioni, lo username e la password dell'utente, al fine di consentire l'autenticazione automatica alla successiva visita del portale.

Impostare un cookie in Java è semplicissimo, tanto più che le classi Action di Struts hanno visibilità tanto della request, javax.servlet.http.HttpServletRequest, quanto della response, javax.servlet.http.HttpServletResponse. Basta creare un cokie, istanza della classe javax.servlet.http.Cookie, impostarne la scadenza in secondi, e aggiungerlo alla response.

```
DesEncrypter encrypter = new DesEncrypter(“secret phrase”));
Cookie usernameCookie = new Cookie("MicroarrayWebUsername",
      encrypter.encrypt(user.getUsername()));
usernameCookie.setMaxAge(60 * 60 * 24 * 30);
```

```
response.addCookie(usernameCookie);
```

Trattandosi di dati sensibili che saranno memorizzati in chiaro sul file system della macchina client, abbiamo preso qualche precauzione criptandone il contenuto tramite un algoritmo di tipo DES, Data Encryption Standard, generato con l'ausilio del package javax.crypto. La frase segreta utilizzata dall'algoritmo DES è stata inserita come parametro di configurazione all'interno del resource bundle principale dell'applicazione, cui si ha semplice accesso da ogni Action class.

5.6.2 Log in automatico

Il log in automatico consente all'utente di effettuare il log in senza passare per la pagina di log in, e quindi senza dover compilarne i form. Il sistema risalirà all'identità dell'utente a partire dai cookie inviati con la request ed effettuerà il log in automaticamente.

Questa funzionalità introduce un rischio per la sicurezza del sistema, in quanto consente l'accesso autenticato a chiunque acceda al sistema dal PC di un utente precedentemente autenticato. Per questo motivo abbiamo inserito una checkbox nel form di login per lasciare all'utente la scelta sul se utilizzare o meno questa funzionalità. Conseguentemente è l'utente che sceglie, se memorizzare o no, dei cookie sulla macchina da cui accede al portale.

5.6.2.1 L'interfaccia javax.servlet.Filter

La specifica Java Sevlet 2,3 introduce un nuovo tipo di componente detto filter, in italiano filtro. I Filtri consentono di aggiungere funzionalità specifiche all'intera applicazione web senza dover mettere mano al codice delle servlet che la compongono. Essi intercettano dinamicamente le request e le response HTTP per trasformare o utilizzare le informazioni in esse contenute, ad esempio per eseguire operazioni di logging o controlli di sicurezza. In generale un

filter non crea esso stesso delle nuove response ma è in grado di eseguire elaborazioni per trasformare le response prodotte da una servlet, ad esempio comprimendone il contenuto per migliorare le performance.

La configurazione dei filter viene esposta in modo dichiarativo nel deployment descriptor dell'applicazione web, indicando, mediante un URL pattern ,quale categoria di request si vuole intercettare. Così il filter elaborerà il contenuto della request prima che questa raggiunga la servlet a cui era destinata. E' possibile configurare più filtri per l'esecuzione in cascata, in modo da elaborare una stessa request in modo modulare. Così facendo arricchiamo l'applicazione web di funzionalità che altrimenti richiederebbero un intervento su ogni singola servlet.

Un filter è una implementazione dell'interfaccia standard javax.servlet.Filter, la quale prevede tre metodi:

```
public void init()
public void doFilter(ServletRequest request,
      ServletResponse response, FilterChain chain)
public void destroy()
```

Il primo metodo, init(), viene invocato dal web container per inizializzare il filtro. Il secondo, doFilter(), è il metodo invocato dal web container per eseguire l'elaborazione vera e propria. Il terzo metodo, destroy(), sarà invocato dal web container per mettere fuori servizio il filtro.

Nel metodo doFilter() lo sviluppatore può inserire il codice relativo alle operazioni che vuole fare eseguire al filtro sulla request o sulla response che vengono passate in input al metodo. Al termine dell'esecuzione della catena di filtri la request filtrata giungerà alla servlet che ha il compito di gestirla.

5.6.2.2 Il filtro realizzato

Abbiamo realizzato un filter che, a partire dai cookie contenenti username e password, che il client invia come header aggiuntivo di ogni request indirizzata al nostro portale, effettua automaticamente il log in. Il filtro preleverà tutti i cookie contenuti nella request che gli è passata come parametro e, tramite il nome, individuerà quelli contenenti le credenziali dell'utente da loggare con una operazione del tipo seguente.

```
Cookie[] cookies = request.getCookies();
String value = null;
if (cookies != null) {
      for (int i=0; i<cookies.length; i++) {
            if (cookies[i].getName().equals(cookieName)) {
                  value = cookies[i].getValue( );
            }
      }
}
```

Il contenuto dei cookie sarà decriptato utilizzando la frase segreta archiviata nel resource bundle con le istruzioni:

```
DesEncrypter encrypter = new DesEncrypter("secret phrase");
value = encrypter.decrypt(value);
```

Le credenziali decriptate saranno poi utilizzate per effettuare la procedura di log in al sistema analogamente a quanto accade nella Action innescata dalla compilazione manuale del form di log in. L'utilizzo del filtro è stato dichiarato nel deployment descriptor, web.xml, dell'applicazione nel modo che segue.

```
<filter>
      <filter-name>AutomaticLoginFilter</filter-name>
<filter-class>microarray.utils.AutomaticLoginFilter</filter-
      class>
</filter>
<filter-mapping>
      <filter-name>AutomaticLoginFilter</filter-name>
      <url-pattern>/*</url-pattern>
</filter-mapping>
```

5.6.3 Il processo di autorizzazione

Una volta autenticata l'identità dell'utente tramite la procedura di log in, manuale o automatica, le informazioni acquisite saranno utilizzate per l'autorizzazione, vale a dire per definire l'insieme di operazioni cui l'utente ha accesso e, conseguentemente, le operazioni cui l'accesso gli è vietato. Il processo di autorizzazione nella nostra applicazione è stato gestito in due modalità complementari, conseguenza del duplice sistema di sicurezza.

Da un lato abbiamo i tre possibili utilizzatori del sistema, il visitatore anonimo, l'utente registrato e l'amministratore, che avranno accesso a tre diversi gruppi di funzionalità, cioè di Action. Dall'altro lato abbiamo i tre livelli di visibilità degli esperimenti: privato, pubblico e di gruppo, a partire dai quali è possibile restringere la visibilità dei dati ad un sottoinsieme degli utenti registrati o anche al solo utente proprietario. La stessa proprietà dell'esperimento comporta la possibilità, da parte di un unico utente, di rimuoverne o modificarne i dati.

5.6.3.1 I livelli di Accesso alle action

Per risolvere il problema dell'accesso alle funzionalità del sistema da parte dei diversi utilizzatori abbiamo associato ad ogni Action class un livello di accesso. Le Action sono state raggruppate in tre gruppi, distinti per livello di sicurezza:

- **livello -1**, Action pubbliche, cui anno accesso tutti gli utilizzatori del sistema, inclusi i visitatori anonimi;
- **livello 0**, Action riservate agli utenti registrati, e quindi anche agli amministratori;
- **livello 1**, Action di amministrazione, riservate ai soli amministratori;

Prima dell'esecuzione di ogni Action, il sistema verificherà che l'utente abbia i privilegi necessari confrontando il tipo di account

dell'utente col livello della Action stessa. Così i visitatori anonimi avranno accesso alle sole Action di livello -1; gli utenti registrati avranno accesso alle Action dei livelli -1 e 0; gli amministratori avranno accesso a tutte le action: livelli -1, 0 e 1.

Le coppie <nome action, livello di accesso> sono state memorizzate in una HashMap inserita nell'application scope, utilizzando un PlugIn di Struts.

5.6.3.2 L'interfaccia PlugIn di Struts

Spesso una applicazione ha bisogno di inizializzare delle risorse allo start up e poi di rilasciarle allo shut down. In una applicazione standard inizializzare e rilasciare risorse non è un problema perché c'è il pieno controllo del codice eseguito all'avvio e all'uscita. Tuttavia, l'utilizzo di simili funzioni non è semplice con Struts, perché l'ActionServlet viene avviata e chiusa dal web container.

Per aggiungere azioni da compiere allo start up e shut down senza estendere l'ActionServlet, cosa che potrebbe risultare complessa, gli sviluppatori di Struts hanno reso disponibile l'intefaccia org.apache.struts.action.PlugIn. L'interfaccia PlugIn, mostrata a seguire, ha due metodi init() e destroy().

```
public interface PlugIn{
public void destroy();
public void init(ActionServlet servlet,ModuleConfig config)
      throws ServletException;
}
```

Allo start up l'ActionServlet verifica la presenza o meno di PlugIn da chiamare. Se ci sono PlugIn da chimare, il metodo init() di ognuna viene eseguito. In modo simile quando il web container chiede lo shut down dell'ActionServlet, questa prima esegue i metodi destroy di ogni PlugIn configurata.

Per poter utilizzare una PlugIn è necessario dichiararne la presenza nel file di configurazione di Struts, struts-config.xml, utilizzando l'apposito tag <plug-in> e specificando il nome della classe da utilizzare. E' possibile utilizzare l'ulteriore tag <setproperty> per passare parametri di inizializzazione alla PlugIn.

5.6.3.3 Il PlugIn realizzato

Utilizzando l'interfaccia PlugIn di Struts abbiamo realizzato una classe che gestisce l'inizializzazione dell'applicazione. Il metodo init() del PlugIn realizzato memorizza nell'application scope i valori di una serie di parametri utili durante l'esecuzione dell'ActionServlet. Come già detto, tra i parametri memorizzati nell'application scope, c'è una HashMap contente le coppie <nome action, livello di accesso> costruita come segue.

```
ServletContext sc = servlet.getServletContext();
HashMap<String, Integer> paths = new HashMap<String,
Integer>();

// Path accessibili agli anonimi
paths.put("/start", new Integer(-1));
paths.put("/login", new Integer(-1));
paths.put("/join", new Integer(-1));
paths.put("/passwordrecovery", new Integer(-1));

// path accessibili agli utenti
paths.put("/myaccount", new Integer(0));
...
paths.put("/analyseslist", new Integer(0));

// path accessibili agli amministratori
paths.put("/userslist", new Integer(1));
...
paths.put("/admintools", new Integer(1));
sc.setAttribute("paths",paths);
```

Il PlugIn è stato configurato nello struts-config.xml utilizzando la sintassi XML seguente.

```
<plug-in className="microarray.plugins.StartupManager"/>
```

5.6.3.4 Il metodo processPreprocess()

Il RequestProcessor, una delle classi chiave del controller di Struts, si occupa della gestione di tutte le request HTTP destinate alla nostra applicazione e, per questo, è il posto ideale per gestirne la sicurezza. Così gli sviluppatori di Struts hanno dotato questa classe di un metodo vuoto, non implementato, da utilizzare come punto di estensione. Il metodo in questione si chiama processPreprocess() ed è stato inserito al solo scopo di consentirne l'overriding da parte dello sviluppatore.

```
protected boolean processPreprocess(HttpServletRequest
      request,HttpServletResponse response) {
}
```

Ogni volta che il RequestProcessor processa una request, chiama il metodo processPreprocess. Normalmente nulla accade, perché il metodo è vuoto, ma noi ne abbiamo effettuato l'overriding per gestire la sicurezza e determinare se l'utente ha i privilegi necessari ad accedere alla Action richiesta.

Il metodo da noi implementato innanzi tutto recupera dalla sessione l'attributo user, contenente una istanza dell'oggetto UserDTO popolata con i dati personali dell'utente, tra gli altri anche il tipo di account.

```
HttpSession session = request.getSession(true);
UserDTO user = (UserDTO) session.getAttribute("user");
int userlevel = -1;
if (user != null) {
      userlevel = user.getAccounttype();
}
```

Una volta acquisito il livello dell'utente: -1 se è un visitatore anonimo, 0 se è un utente registrato o 1 se è un amministratore, il metodo provvede a individuare il livello della Action richiesta individuata dal path associato alla request.

```
int level = -1;
HashMap paths = (HashMap)session.getServletContext().
```

```
      getAttribute("paths");
Integer accessLevel =
      (Integer)paths.get(RequestedActionPath);
if (accessLevel != null) {
      level = accessLevel.intValue();
}
```

A questo punto sarà possibile scegliere se continuare a processare la request, perché l'utente è dotato dei privilegi necessari, o reindirizzare l'utente verso la action forbidden, che gli notifica l'impossibilità di eseguire l'operazione richiesta.

5.6.3.5 Gestione dei livelli di visibilità

Con quanto visto in precedenza abbiamo filtrato l'accesso degli utenti alle funzionalità del portale, sulla base dei privilegi detenuti da ognuno, utente, amministratore o semplice visitatore anonimo. Ora affrontiamo il problema della visibilità dei dati, filtrando, ad un livello più fine, le operazioni richieste dall'utente.

Con l'implementazione del metodo processPreprocess() abbiamo consentito, ad esempio, ai soli utenti registrati e amministratori l'accesso alla lista degli esperimenti. Ora bisogna stabilire, in base al livello di visibilità, quali esperimenti l'utente può visualizzare. Se l'utente è un amministratore, potrà ovviamente visualizzare tutti gli esperimenti archiviati nel database, ma se è un semplice utente dovrà vedere:

- tutti gli esperimenti con visibilità pubblica;
- quegli esperimenti con visibilità di gruppo pubblicati da un utente appartenente al suo stesso gruppo;
- i soli esperimenti con visibilità privata di cui è proprietario.

Per ottenere che ogni utente, in base al gruppo di appartenenza e al livello del proprio account, visualizzi solo gli esperimenti di cui sopra, abbiamo operato una personalizzazione della query SQL che si occupa di recuperare gli esperimenti dal database sulla base del-

le informazioni accertate leggendo l'attributo user memorizzato nella sessione.

A questo punto occorre scendere ad un livello di dettaglio ancora più basso. Degli esperimenti che l'utente può vedere, e di cui può scaricare i dati, l'utente potrà cancellare o modificare solo quelli di sua proprietà. Per questo è stata creata una classe DataAccessBean, dotata di metodi del tipo seguente.

```
boolean userCanAdd(int experiment_id, HttpSession    session)
boolean userCanDelete(int experiment_id, HttpSession
      session)
boolean userCanUpdate(int experiment_id, HttpSession
      session)
boolean userCanView(int experiment_id, HttpSession   session)
HashMap<String, Boolean> userCanView(ArrayList<String>
      experiments, HttpSession session)
boolean userCanDownload(int experiment_id, HttpSession
      session)
boolean isAdmin(HttpSession session)
```

Tali metodi in generale restituiscono un booleano che vale true, se l'utente può effettuare l'azione, o false se non può. Così ogni volta che l'utente accederà ad una Action di modifica o cancellazione di un dato, sarà verificata l'esistenza del diritto a compiere quell'azione.

Precisiamo che questa ulteriore operazione di filtraggio degli accessi non si limita alla fase di preparazione di quello che l'applicazione presenta nelle proprie pagine, ad esempio pulsanti abilitati o disabilitati, ma si estende anche ai tentativi di accesso dei malintenzionati tramite compilazione diretta dell'url.

5.6.4 Protezione delle pagine JSP

Un ultimo problema di sicurezza riguarda l'accesso alle pagine JSP. Potrebbe essere utile, infatti, vietare l'accesso diretto a queste che è ottenibile, ad esempio, tramite la digitazione dell'indirizzo nella barra dell'URL del browser. I maleintenzionati, pur non avendo accesso alle Action class, e quindi alle funzionalità dell'applicazione, potrebbero trarre dalle pagine informazioni riservate o comunque utili a fini illeciti.

Per ridurre questo rischio, è bastato spostare le pagine JSP in una directory interna a WEB-INF. Sulla base delle specifiche Servlet, infatti, WEB-INF non è parte dell'area pubblica di una applicazione web. Perciò nessuna risorsa interna a WEB-INF può essere acceduta direttamente da un client. Ad ogni modo sarà possibile passare al client le pagine così archiviate, ma il client non potrà più richiedere l'accesso diretto a queste: abbiamo protetto il nostro sito dagli accessi indesiderati, pur potendo comporre la view utilizzando le stesse pagine JSP.

L'utilizzo di questa tecnica comporta le redirezione dei forward alle pagine nella nuova locazione. Quindi dovremo trasformare come segue il contenuto del file struts-config.xml:

```
<action path="/logout" type="microarray.actions.LogoutAction"
      validate="false">
      <forward name="logout" path="/WEB-INF/jsp/index.jsp"/>
</action>
```

Anche se, in realtà, l'utilizzo di Struts Tiles, illustrato in seguito, ci ha imposto simili modifiche nel file contenente le definition, invece che nel file di configurazione di Struts.

Uno svantaggio di questa tecnica, per altro poco seccante, sta nel fatto che l'accesso alle pagine debba sempre avvenire tramite una Action class, anche nei casi più banali. Un secondo problema potrebbe sorgere qualora si utilizzi un servlet container che non implementi le particolari specifiche Servlet a noi necessarie, ma non è il caso di Tomcat.

5.7 Invio di e-mail

Una applicazione web spesso ha bisogno di notificare agli amministratori, o agli utenti, eventi con una e-mail. Il sistema sviluppato, ad esempio, ha utilizzato questa funzionalità per notificare all'amministratore ogni nuova registrazione di un utente, in modo che questo possa verificarne l'identità, la leggittima appartenenza ad un gruppo di ricerca, ed infine sbloccarne lo stato.

Abbiamo sfruttato la possibilità di inviare e-mail anche per mandare un log del processo di indicizzazione batch all'amministratore e, eventualmente, per notificare problemi di elaborazione all'utente che abbia inserito dati non indicizzabili. In questo modo, qualora si presentino problemi nel processo di indicizzazione batch, che, lo ricordiamo, viene schedulato quotidianamente in un orario di basso carico del sistema, abbiamo ritenuto di accelerare l'interazione utente-amministratore per l'analisi dell'errore incontrato.

Una e-mail viene inviata, infine, per ricordare le password dimenticate. Qualora un utente registrato abbia dimenticato la propria password, può accedere alla pagina di recupero password, disponibile nell'area pubblica del portale, ed inserire nell'apposito form il proprio username. Il sistema, utilizzando lo username che, ricordiamo, è un identificatore dell'entità utente, risalirà alla passeword e la invierà all'indirizzo email fornito all'atto della registrazione.

Anche in questo caso, abbiamo implementato la funzionalità utilizzando un prodotto open source di Apache: il package Commons Email. L'invio di un semplice messaggio di testo può avvenire per mezzo della classe SimpleEmail; la classe MultiPartEmail consente invece l'invio di messaggi di posta contenenti allegati, mentre la classe HtmlEmail consente l'invio di messaggi di posta contenenti un testo formattato con HTML, oltre che di allegati.

Commons Email semplifica l'invio di un messaggio di posta fino a renderlo banale. Abbiamo configurato i parametri necessari all'invio di un messaggio di posta nel file ApplicationResources.properties, abusando così del resource bundle di Struts.

```
...
config.email.hostname=mailer.isa.cnr.it
config.email.username=username@isa.cnr.it
config.email.password=*******
config.email.recipient=dacierno.a@isa.cnr.it
config.email.recipient.name=Antonio D'Acierno
config.email.sender=dacierno.a@isa.cnr.it
config.email.sender.name=Microarray Web
config.email.subject=New user joined Microarray
config.email.message=User required to join Microarray...
config.email.passwordrecovery.message=Your password is...
config.email.passwordrecovery.subject=Password reminder
...
```

L'accesso al resource bundle, avviene, al solito utilizzando il metodo getResources(HttpServletRequest request) della classe Action. Una volta recuperati i parametri dal resource bundle, per inviare un messaggio occorre creare un oggetto SimpleEmail, inizializzarlo con i parametri necessari e, infine, invocare il metodo send().

```
SimpleEmail email = new SimpleEmail();
email.setHostName(hostname);
if (!username.equals("") && !password.equals("")) {
      email.setAuthentication(username, password);
}
email.addTo(recipient, recipientName);
email.setFrom(sender, senderName);
email.setSubject(subject);
email.setMsg(message);
email.send();
```

5.8 Compressione dei dati

La grossa mole di dati da memorizzare nel database, unitamente alla ridondanza introdotta per l'utilizzo di un motore di ricerca, ci hanno spinto a considerare la possibilità di comprimere on the fly i dati, prima della memorizzazione sotto forma di blob nel database. In questo modo, pur memorizzando due volte lo stesso dato, una volta nell'indice e una volta nel database, lo spazio complessivamente occupato sarebbe stato meglio gestibile.

Il principale svantaggio dell'utilizzo di questa tecnica sarebbe stato la necessità di ricorrere ad ulteriori risorse computazionali, principalmente cpu e memoria, durante il già complesso processo di archiviazione dei dati. Tuttavia, il vantaggio prestazionale che avremmo ottenuto nella gestione del database, ci ha indotti a testare questa soluzione. Per le prove abbiamo utilizzato il package java.util.zip, che fornisce classi per la gestione dei comuni formati ZIP e GZIP, oltre ad una implementazione dell'algoritmo standard DEFLATE per la compressione negli stessi formati.

Dalle prove effettuate sull'insieme di dati a nostra disposizione è risultata la riduzione della dimensione dei file di testo inseriti ad 1/4 o 1/5 di quella di partenza, a fronte di un allungamento del tempo necessario all'archiviazione di 4-5 secondi. Tenuto conto del fatto che il tempo di attesa per l'archiviazione è largamente influenzato dal trasferimento dei dati su rete e, eventualmente, dalla richiesta di indicizzazione immediata, abbiamo ritenuto di scarso impatto l'ulteriore attesa per la compressione.

Abbiamo accettato, quindi, di impegnare maggiori risorse computazionali durante il processo di indicizzazione, per ottenere un vantaggio sotto forma di risparmio dello spazio su disco. Abbiamo rilevato, inoltre, una migliore reattività del DBMS nell'esecuzione di query sulle tabelle interessate. Non meno interessante è la riduzio-

ne delle risorse hardware minime necessarie a far girare l'applicativo, essendo lo sforzo computazionale richiesto alla portata delle CPU più economiche, mentre l'occupazione di uno spazio minore eviterebbe, al crescere dei dati archiviati, il ricorso a un sottosistema disco dedicato.

A seguire riportiamo il metodo utilizzato per la compressione in un array di byte dei dati disponibili nello stesso formato, omettendo, per semplicità, la parte relativa alla gestione delle eccezioni.

```
public byte[] compress(byte[] decompressedData) {
      // Create the compressor with highest
      // level of compression
      Deflater compressor = new Deflater();
      compressor.setLevel(Deflater.BEST_COMPRESSION);

      // Give the compressor the data to compress
      compressor.setInput(decompressedData);
      compressor.finish();

      // Create an expandable byte array to
      // hold the compressed data.
      ByteArrayOutputStream bos = new ByteArrayOutputStream
            (decompressedData.length);

      // Compress the data
      byte[] buf = new byte[1024];
      while (!compressor.finished()) {
            int count = compressor.deflate(buf);
            bos.write(buf, 0, count);
      }
      bos.close();

      // Get the compressed data
      byte[] compressedData = bos.toByteArray();
      return compressedData;
}
```

Il metodo utilizzato per la decompressione, sempre di un array di byte in un array di byte, invece, è il seguente:

```
public byte[] decompress(byte[] compressedData) {
      // Create the decompressor and give it
      // the data to compress
      Inflater decompressor = new Inflater();
```

```
        decompressor.setInput(compressedData);

        // Create an expandable byte array
        // to hold the decompressed data
        ByteArrayOutputStream bos = new
              ByteArrayOutputStream(compressedData.length);

        // Decompress the data
        byte[] buf = new byte[1024];
        while (!decompressor.finished()) {
              int count = decompressor.inflate(buf);
              bos.write(buf, 0, count);
        }
        bos.close();

        // Get the decompressed data
        byte[] decompressedData = bos.toByteArray();
        return decompressedData;
}
```

5.9 Prevenzione dei submit multipli di uno stesso form

Il problema della doppia sottomissione dei form sorge quando un utente clicca più di una volta sul pulsante submit di uno stesso form prima di ricevere la response, o quando sottomette un form cui ha acceduto premendo il pulsante back del browser. Infatti, se il server impiega qualche secondo per processare una request, l'utente, pensando che il primo submit non abbia avuto effetto, potrebbe essere erroneamente indotto a cliccare nuovamente sul pulsante submit, generando di fatto un nuovo submit dello stesso form.

Ciò può portare i dati in uno stato inconsistente e deve essere evitato. Struts fornisce una soluzione a questo problema tramite i metodi saveToken() e isTokenValid() della classe Action. Il metodo saveToken(HttpServletRequest request) salva un token, una stringa alfanumerica identificativa della transazione avviata, nella sessione dell'utente. Il metodo isTokenValid() confronta il token inviato con la request a quello salvato nella sessione, consentendo così di stabilire se il form è stato sottomesso più di una volta.

Per far ciò è necessario che la pagina JSP contenente il form sia generata da una Action che invochi il metodo saveToken(). La pagina così generata conterrà un input aggiuntivo di tipo hidden in cui è inserito il valore del token. Così, al submit del form, il valore del token sarà inviato come parametro della request alla Action deputata a gestirla. Questa seconda Action potrà invocare il metodo isTokenValid() per confrontare il token ricevuto con la request con quello salvato nella sessione dalla prima Action.

A seguire riportiamo un semplice esempio dell'utilizzo di questa tecnica nell'ambito di una Action class che estende DispatchAction. Il primo metodo crea il token e lo salva nella sessione, il secondo processa la request solo se il token è valido.

```
public  class PurchaseOrderAction extends DispatchAction {

public ActionForward load(ActionMapping mapping,
      ActionForm form,
      HttpServletRequest  request,
      HttpServletResponse response) throws Exception {
            // Save the token
            saveToken(request)
            // Code for loading the form
      }

public ActionForward submitOrder(ActionMapping mapping,
      ActionForm form,
      HttpServletRequest  request,
      HttpServletResponse response) throws Exception {
            // Check the token.
            // Proceed only if token is valid
            if(isTokenValid(request,true)) {
                  //implement submit functionality
            } else {
                  return mapping.findForward("failure");
            }
      }
}
```

5.10 I18n, l'internazionalizzazione

Il mondo di oggi è più piccolo di quello di dieci anni fa, e il mondo di domani sarà ancora più piccolo. Questa restrizione dei confini è in parte dovuta all'espansione del Web come sistema di comunicazione per i cittadini e le aziende del mondo intero. L'internazionalizzazione, in inglese internationalization o, per i più pigri I18N, è la tecnica con cui si progetta un software in grado di supportare linguaggi multipli senza che sia necessario preparare una nuova applicazione per gestire ogni lingua.

Le informazioni da internazionalizzare non si limitano, in realtà alla sola lingua, ma si estendono al formato delle date, dell'ora, dei numeri, della valuta e, più in generale, ad ogni entità che può assumere un aspetto diverso al variare del locale, cioè del luogo in cui si trova l'utente. Struts basa il proprio supporto all'internazionalizzazione sulle funzionalità offerte dalla piattaforma Java, con la restrizione ai testi e alle immagini da utilizzare per costruire la view.

Gli oggetti su cui Struts basa l'internazionalizzazione sono i seguenti:

- la classe java.util.Locale. Ogni Locale rappresenta una particolare scelta di posizione geografica e lingua ed implica un insieme di assunzioni sulla formattazione da utilizzare per entità quali i numeri e le date;
- la classe java.util.ResourceBundle che fornisce gli strumenti fondamentali per gestire messaggi in molteplici lingue;
- La classe org.apache.struts.util.MessageResources che consente di trattare un insieme di resource bundle come un database, e di recuperare in modo trasparente il particolare messaggio per il Locale associato all'utente.

Per internazionalizzare una applicazione, per prima cosa bisogna preparare un insieme di semplici file di testo con estensione properties. Ognuno di essi conterrà una coppia key/value, cioè chiave/valore, per ogni messaggio che la nostra applicazione dovrà presentare, in una lingua specifica. E' centrale il formato del nome del file, che sarà ApplicationResources.properties per il file contenente i messaggi nella lingua di default, solitamente l'inglese; avrà il nome seguito dai due caratteri ISO che identificano le lingue nazionali nel caso delle altre lingue, ad esempio:

- ApplicationResources_de.properties per il tedesco;
- ApplicationResources_fr.properties per il francese;
- ApplicationResources_it.properties per l'italiano;
- ApplicationResources_es.properties per lo spagnolo.

Struts preparerà la view utilizzando il i messaggi immagazzinati nel file corrispondente alla lingua del client, se questo esiste, o quelli immagazzinati nel file di default, se la lingua non è supportata.

Prima di poter utilizzare il supporto all'I18N occorre definire, nel file di configurazione di Struts, uno o più MessageResources per la nostra applicazione, o per un modulo di essa. E' anche possibile utilizzare differenti bundles per la stessa applicazione, utilizzando una key per identificare ognuno. Un esempio della sintassi utilizzata è il seguente:

```
<message-resources null="false"
      parameter="ApplicationResources"/>
```

Dove l'attributo obbligatorio “parameter” indica il nome del resource bundle di default, mentre l'attributo “null” si riferisce alla possibilità o meno di stampare messaggi del tipo ???key??? invece che null, quando non c'è un messaggio associato a key nel resource bundle.

Abbiamo preparato per la nostra applicazione due resource bundle, uno di default per l'inglese e uno per l'italiano che contengono rispettivamente messaggi del tipo:

```
home.msg=Welcome to Microarray Web App!
login.msg=Click here to log in!
```

oppure

```
home.msg=Benvenuti in Microarray Web App!
login.msg=Clicca qui per entrare!
```

E'possibile accedere ai messaggi utilizzando l'API di Struts o le tag libraries. Ad esempio, per la stampa di un messaggio è possibile utilizzare il tag message della libreria bean.

```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>

<bean:message key="home.msg" />
```

All'interno di una Action class, ivece, è possibile accedere ai messaggi con una sintassi del tipo:

```
MessageResources prop = getResources(request);
String msg =props.getMessage("home.msg");
```

5.11 Validazione dei dati in ingresso

Per la validazione lato server dei dati immessi in un form HTML, Struts mette a disposizione l'overriding del metodo validate() della classe org.apache.struts.action.ActionForm. Il RequestProcessor chiama questo metodo immediatamente prima di chiamare il metodo execute() della Action class. Nel caso in cui il formato dei dati non corrisponda a quello previsto, il framework provvede a visualizzare nuovamente la pagina con i dati immessi, includendo messaggi di errore, affinchè l'utente possa apprendere quali errori ha commesso.

Questo approccio al problema della validazione, pur alleviando il compito dello sviluppatore, presenta alcuni inconvenienti. L'inconveniente principale nasce quando in più form della stessa applicazione occorra effettuare una validazione basata sulla stessa regola. Dalla moltiplicazione del codice che ne risulta, un cambiamento in una regola di validazione implica una modifica ai tutti i sorgenti che utilizzano quel frammento di codice.

Il superamento di questa tecnica si realizza con l'utilizzo di un PlugIn incluso nella distribuzione di Struts: Validator Framework. Si tratta di un framework che fornisce gli strumenti per effettuare in modo automatico e dichiarativo i controlli formali sui campi di un form HTML. L'utilizzo di Validator, creando i FormBean come estensione di org.apache.struts.validator.ValidatorForm in luogo di ActionForm, automatizza la validazione ed elimina la necessità di compilare un metodo validate() per ogni form.

Validator include un insieme di metodi in grado di eseguire i più comuni controlli di validazione ed include la possibilità di creare routine custom per i controlli che il framework non include. Inoltre esso offre la possibilità di validare i dati anche lato client, mediante

l'utilizzo di metodi JavaScript che replicano le funzionalità offerte lato server.

La configurazione di Validator avviene in due file XML: validator-rules.xml e validation.xml, quindi esternamente al codice applicativo. Nel file validator-rules.xml bisogna dichiarate tutte le routine di validazione utilizzate, i loro nomi logici e il codice JavScript corrispondente a ciascuna di essa per la validazione client-side. Nel file validation.xml, invece, si specifica quali form validare, identificandoli con lo stesso nome definito nel file di configurazione di Struts, quali routine di validazione applicare e come applicarle.

In sintesi, per utilizzare Validator framework, bisogna eseguire i seguenti step:

- dichiarare l'utilizzo del PlugIn Validator nel file di configurazione di Struts;
- preparare i due file di configurazione, validator-rules.xml e validation.xml
- creare i FormBean estendendo la classe ValidatorForm

Per configurare il PlugIn e fare in modo che Struts lo carichi all'avvio dell'applicazione, abbiamo utilizzato la sintassi:

```
<plug-in
className="org.apache.struts.validator.ValidatorPlugIn">
<set-property property="stopOnFirstError" value="false"/>
<set-property property="pathnames"
value="/WEB-INF/validator-rules.xml,/WEB-
INF/validation.xml,/WEB-INF/validation-extends-rule.xml"/>
</plug-in>
```

specificando il nome esteso della classe ValidatorPlugIn, i path dei file di configurazione utilizzati, e una proprietà con la quale chiediamo che Validator continui la validazione anche dopo aver individuato il primo errore.

Normalmente non è necessario apportare modifiche al file validator-rules.xml, perché questo include le regole di validazione più comuni, tuttavia, avendo implementato due ulteriori metodi di validazione, abbiamo aggiunto quanto segue.

```
<validator name="twofields"
classname="microarray.plugins.StrutsValidator"
method="validateTwoFields"
methodParams="java.lang.Object,
org.apache.commons.validator.ValidatorAction,
org.apache.commons.validator.Field,
org.apache.struts.action.ActionMessages,
javax.servlet.http.HttpServletRequest,
javax.servlet.ServletContext"
msg="errors.twofields"/>

<validator name="filetype"
classname="microarray.plugins.StrutsValidator"
method="validateFileExtension"
methodParams="java.lang.Object,
org.apache.commons.validator.ValidatorAction,
org.apache.commons.validator.Field,
org.apache.struts.action.ActionMessages,
javax.servlet.http.HttpServletRequest,
javax.servlet.ServletContext"
msg="errors.filetype"/>
```

La prima routine, twofields, serve a controllare che l'utente, quando sceglie la propria password, compili con lo stesso valore i due campi che gli vengono proposti. La seconda routine, filetype, effettua una semplice validazione dell'estensione dei file di cui l'utente fa l'upload verso il sistema.

Per entrambe sono stati specificati: il nome della classe che contiene il metodo di validazione; il nome del metodo e i parametri che gli saranno passati. E' stato specificato, inoltre, un attributo msg che indica a Validator quale messaggio prelevare dal resource bundle quando si riscontra un errore di formato. Ciò testimonia la possibilità di internazionalizzare i messaggi utilizzati dal framework di validazione con la tecnica illustrata nei paragrafi precedenti.

Nel file validation.xml specificheremo a quali form applicare Validator e con quali regole. A seguire riportiamo la sintassi utilizzata per l'ActionForm loginForm, da compilare per il log in al sistema.

```
<form name="loginForm">
<field depends="required,maxlength" property="username">
<arg key="userstable.username" name="required" position="0"/>
<arg key="userstable.username" name="minlength"
      position="0"/>
<arg key="userstable.username" name="maxlength"
      position="0"/>
<arg key="${var:maxlength}" name="maxlength" position="1"
      resource="false"/>
<var>
<var-name>maxlength</var-name>
<var-value>25</var-value>
</var>
</field>
<field depends="required,maxlength" property="password">
<arg key="password.required" name="required" position="0"/>
<arg key="password.required" name="maxlength" position="0"/>
<arg key="${var:maxlength}" name="maxlength" position="1"
      resource="false"/>
<var>
<var-name>maxlength</var-name>
<var-value>15</var-value>
</var>
</field>
</form>
```

La regola di validazione dichiarata impone che i field username e password di loginForm siano entrambi obbligatori (regola required) e che debbano avere una lunghezza massima (regola maxlength) rispettivamente di 25 e 15 caratteri. Sono stati dichiarati inoltre degli argomenti per la sostituzione ai segnaposto, identificati dal valore dell'attributo position, introdotti nei messaggi raccolti dal resource bundle.

```
errors.required={0} is required.
errors.maxlength={0} can not be greater than {1} characters.

userstable.username=Username
userstable.password=Password

username.required=Username
```

```
password.required=Password
```

Infine, per quanto riguarda la validazione lato client, questa può essere introdotta utilizzando il tag javascript della libreria HTML distribuita con Struts. Ad esempio:

```
<html:javascript formName="loginForm"/>
```

invoca l'utilizzo della validazione client-side per loginForm. La validazione con JavaScript integra la validazione lato server, consentendo di evitare inutili round-trip con il server dovuti a successivi submit del form per via di errori di validazione sui campi compilati.

5.12 Tuning del processo di indicizzazione

Quando si utilizza Lucene per indicizzare piccole collezioni di documenti non è necessario, in generale, alterare le impostazioni di default del processo di indicizzazione. In una applicazione come la nostra, invece, che, in ragione delle scelte progettuali adottate, indicizza ogni volta centinaia di migliaia di Document è necessario tenere sotto più stretto controllo tale processo. Un buon tuning del processo di indicizzazione, infatti, può portare al raggiungimento di considerevoli incrementi prestazionali.

Lucene utilizza un certo numero di parametri per definire le risorse da utilizzare in fase di indicizzazione. Agendo su questi parametri, ad esempio, nel caso in cui si abbia a disposizione una macchina dotata di memoria RAM inutilizzata, metteremo al corrente Lucene della possibilità di utilizzare quella risorsa.

Normalmente il collo di bottiglia del processo di indicizzazione sta nella scrittura dei file che costituiscono l'indice sul disco. In effetti, nelle prove effettuate utilizzando un disco da 5400rpm, il processo di indicizzazione è risultato molto più lento, rispetto al caso in cui si sia utilizzato un disco da 7200rpm. Per questo occorre istruire attentamente Lucene su come e quando effettuare le scritture su disco durante l'indicizzazione di nuovi documenti e la modifica degli indici esistenti.

Quando aggiunge nuovi Document all'indice, Lucene scrive in un buffer nella memoria centrale, invece di effettuare immediatamente la scrittura definitiva su disco. Questo buffering ha motivi prestazionali e, fortunatamente, la classe IndexWriter, che si occupa della scrittura dell'indice, espone delle variabili che consentono di stabilire la dimensione del buffer stesso e la frequenza delle scritture su

disco. Le variabili su cui si agisce più frequentemente sono riportate nella tabella a seguire con la rispettiva descrizione degli effetti sul processo di indicizzazione.

Variabile	Default	Descrizione
mergeFactor	10	Determina quanto spesso il metodo addDocument() fonderà segmenti di indice. A valori più bassi corrisponderà l'utilizzo di una minore quantità di RAM e una maggiore velocità della ricerca su indici non ottimizzati, ma l'indicizzazione sarà più lenta. A valori più alti corrisponderà un maggiore utilizzo di RAM e una minore velocità delle ricerche su indici non ottimizzati, ma l'indicizzazione sarà più veloce. Così valori più alti (>10) saranno migliori per l'indicizzazione batch, mentre valori più bassi (<10) daranno prestazioni migliori nell'indicizzazione incrementale.
maxMergeDocs	Integer. MAX_VALUE	Determina il massimo numero di Document di cui il metodo addDocument() effettua il merge. Valori più bassi (<10000) sono migliori per l'indicizzazione incrementale perchè ciò limita la durata delle pause durante l'indicizzazione. Valori più alti (>10000) sono utili all'indicizzazione batch e causano un incremento nelle prestazioni della ricerca.
maxBufferedDocs	0 (automatico)	Determina il numero minimo di Document richiesti prima che il processo di indicizzazione effettui il flush in un segmento sul disco di quanto contenuto nel buffer in RAM. A valori più grandi corrisponde una indicizzazione più rapida. Quando questa variabile è impostata, l'IndexWriter farà il flush basandosi sul numero impostato invece che sull'utilizzo della RAM.

		Il valore 0 corrisponde a una frequenza di flush basata sull'utilizzo della RAM.
maxFieldLength	10000	Il massimo numero di termini che saranno indicizzati per un singolo Field di un Document. Questo limita l'ammontare di memoria richiesta per l'indicizzazione, cosicché collezioni di testi molto corposi non causino il crash del processo per un eccessivo utilizzo di memoria. Questa impostazione si riferisce al numero di termini ricorrenti, non al numero di termini diversi.

A queste variabili si aggiungono le variabili d'ambiente relative alla Java Virtual Machine. Infatti, utilizzando la piattaforma Java, la quantità di memoria disponibile per la nostra applicazione è limitata da quanto viene impostato all'avvio della JVM, piuttosto che da quanta memoria è fisicamente disponibile. Comunque, analizzate le risorse a disposizione sulla macchina server utilizzata, l'impostazione di queste variabili consente di ottimizzare le performance ottenute.

5.13 OpenSymphony Quartz

Quartz, prodotto open source di OpenSymphony, è un framework per la schedulazione integrabile, o almeno utilizzabile, da qualsiasi applicazione J2EE o J2SE, sia per necessità banali che per la gestione di decine di migliaia di task. Abbiamo utilizzato Quartz, sfruttando un ristrettissimo insieme delle funzionalità offerte, per la schedulazione del processo di indicizzazione batch.

Il nostro obbiettivo era lanciare automaticamente, in orario notturno, il processo di indicizzazione batch, in modo da ottenere il quotidiano aggiornamento del contenuto dell'indice. Per questo siamo andati alla ricerca di una soluzione semplice, ma che consentisse di ottenere rapidamente quanto auspicato. Una rapida ricerca sul Web ci ha convinti che l'utilizzo di Quartz avrebbe consentito di schedulare il nostro processo batch con semplicità, senza stravolgere quanto già sviluppato.

Nel linguaggio utilizzato dagli sviluppatori di Quartz, un Job è semplicemente una classe Java che esegue un compito, un task. Non importa in che modo sia codificato il task, il solo prerequisito è che si implementi l'interfaccia org.quartz.Job, sollevando una eccezione di tipo JobExecutionException in caso di errori. L'interfaccia Job contiene un unico metodo, il metodo execute(), che sarà implementato dallo sviluppatore includendovi il codice che esegue effettivamente il task.

```
package org.quartz;

public interface Job {
      public void execute(JobExecutionContext context)
      throws JobExecutionException;
}
```

I Job sono solo un lato della medaglia. Gli sviluppatori di Quartz hanno infatti scelto di separare il concetto di Job da quello di schedule, la lista degli eventi e degli orari in cui dovranno avvenire. Quartz utilizza invece un Trigger, un grilletto, per indicare allo Scheduler quando un lavoro debba essere schedulato, o fired, sparato. Il framework include una manciata di Trigger predefiniti, utili nei compiti più comuni; i due più utili sono sicuramente il SimpleTrigger e il CronTrigger.

Il SimpleTrigger è progettato per sopperire al bisogno di lanciare semplicemente un Job in un dato istante e ripeterne n volte l'esecuzione, magari attendendo un certo numero di secondi tra l'una e l'altra. Il CronTrigger, invece, può essere utilizzato per schedulare Job periodici, ad un certo orario o ogni x intervalli di tempo. Consente, ad esempio, l'esecuzione di un Job tutti i giorni ad un certo orario, eccetto il Venerdì e il Sabato.

Ogni qual volta si vorrà aggiungere un Job allo Scheduler, il nostro programma dovrà creare l'oggetto JobDetail, che conterrà un insieme di informazioni sul Job da schedulare, e un JobDataMap, che sarà utilizzato per immagazzinare informazioni di stato per una certa istanza della classe Job, ad esempio un riferimento al DataSource utilizzato nel Job.

Il primo passo da compiere per integrare Quartz in una applicazione web, è la creazione di un nuovo Scheduler quando l'application container avvia la nostra applicazione. Abbiamo già visto che, per effettuare elaborazioni all'avvio dell'applicazione, Struts utilizza il meccanismo dei PlugIn. Per questo, il primo passo per utilizzare Quartz è configurare un nuovo PlugIn nel file di configurazione struts-config.xml.

```
<plug-in className="microarray.plugins.SchedulerPlugIn">
      <set-property property="startupDelay" value="0"/>
      <set-property property="startOnLoad" value="false"/>
</plug-in>
```

Il secondo passo è configurare, nel file web.xml dell'applicazione, l'avvio della servlet org.quartz.ee.servlet.QuartzInitializerServlet che provvede ad aggiungere l'oggetto SchedulerFactory al ServletContext in modo che sia possibile accedere ad esso dai PlugIn di Struts. Lo SchedulerFactory, a sua volta, provvederà a fornirci lo Scheduler nel codice del PlugIn.

```
<servlet>
<servlet-name>QuartzInitializer</servlet-name>
<servlet-
class>org.quartz.ee.servlet.QuartzInitializerServlet</servlet
-class>
<init-param>
<param-name>shutdown-on-unload</param-name>
<param-value>true</param-value>
</init-param>
<init-param>
<param-name>start-scheduler-on-load</param-name>
<param-value>true</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
```

Il parametro di inizializzazione “shutdown-on-unload” serve a specificare la richiesta che il metodo scheduler.shutdown(), che dealloca le risorse, sia chiamato quando la servlet viene distrutta (di solito quando si spegne l'application server). Il parametro di inizializzazione “start-scheduler-on-load” richiede, invece, che il metodo scheduler.start(), che fa partire lo scheduler, sia chiamato automaticamente quando la servlet viene caricata per la prima volta.

L'ultimo passo configurativo da compiere è la creazione del file quartz.properties nella cartella WEB-INF/classes. Questo file conterrà semplicemente quanto segue:

```
#========================================
# Configure Main Scheduler Properties
#========================================
org.quartz.scheduler.instanceName = MyScheduler
org.quartz.scheduler.instanceId = AUTO

#========================================
# Configure ThreadPool
```

```
#========================================
org.quartz.threadPool.class =
      org.quartz.simpl.SimpleThreadPool
org.quartz.threadPool.threadCount = 12
org.quartz.threadPool.threadPriority = 5
```

cioè un insieme di paramtetri di configurazione che lo SchedulerFactory utilizzerà per creare lo Scheduler.

Il metodo init() del PlugIn da noi realizzato conterrà semplicemente il codice che, ottenuta una istanza dello Scheduler, gli affiderà il compito di lanciare quotidianamente il Job che realizza l'indicizzazione batch, così come segue.

Prima di tutto occorrerà recuperare dal ServletContext una istanza dello SchedulerFactory:

```
StdSchedulerFactory factory =
(StdSchedulerFactory)ctx.getAttribute(QuartzInitializerServle
      t.QUARTZ_FACTORY_KEY);
```

Otterremo il nostro Scheduler dallo SchedulerFactory e lo avvieremo con le istruzioni:

```
Scheduler scheduler = factory.getScheduler();
scheduler.start();
```

Creeremo l'oggetto JobDetail per il Job di indicizzazione batch associando a questo un riferimento al DataSource da utilizzare.

```
JobDetail jobDetail = new JobDetail("IndexingJob", null,
      microarray.utils.IndexingJob.class);

// Set DataSource in JobDataMap
JobDataMap map=new JobDataMap();
map.put("DataSource",
      (DataSource)ctx.getAttribute("microarray2"));
jobDetail.setJobDataMap(map);
```

Creeremo un Trigger giornaliero che si attiverà alle due di ogni notte:

```
// Trigger that fire every day at 02:00 am
Trigger nightlyTrigger = TriggerUtils.makeDailyTrigger(2,0);
```

```
// Set start time to now
nightlyTrigger.setStartTime(new Date());
nightlyTrigger.setName("nightlyTrigger");
```

Infine utilizzeremo lo Scheduler per schedulare il Job di indicizzazione mediante il Trigger definito:

```
scheduler.scheduleJob(jobDetail, nightlyTrigger);
```

A questo punto alle due di ogni notte sarà lanciato il nostro processo di indicizzazione batch.

5.14 Struts Tiles e CSS

Struts Tiles è un templating system, cioè un sistema per la gestione di template. Un template è una particolare pagina JSP che utilizza dei tag per descrivere il layout, cioè la struttura, di una pagina. Esso descrive l'aspetto della pagina senza specificarne il contenuto, che sarà definito solo a runtime. L'obiettivo è utilizzare lo stesso layout per più pagine della nostra applicazione web

L'utilizzo di Tiles permette di definire il layout di una pagina assemblando le cosidette tile, il cui significato è banalmente mattonelle, in modo che le stesse tile possano essere riciclate per costruire diverse strutture. Nella maggior parte dei casi un tile non è altro che una pagina JSP, ma potrebbe anche essere costituito a sua volta da un insieme di tile, realizzando così una struttura ad albero. Questa tecnologia, alternativa all'utilizzo degli “include”, separando i contenuti dal layout, cancella ogni duplicazione del codice che definisce l'aspetto delle pagine.

Il primo passo nell'utilizzo di Tiles consiste nella definizione di un template, una pagina JSP in cui è definito il layout dell'applicazione. Non potendo riportare, per ragioni di spazio, il template che abbiamo effettivamente utilizzato nell'applicazione, a seguire mostriamo il più semplice template possibile.

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/tiles.tld" prefix="tiles"%>

<html:html>
<head>
<title><tiles:getAsString name="title" /></title>
<html:base/>
</head>

<body>
<!-- Header page information -->
<tiles:insert attribute="header"/>
```

```
<!-- Menu bar -->
<tiles:insert attribute="menu"/>

<!-- Main body information -->
<tiles:insert attribute="body"/>

<!-- Footer page information -->
<tiles:insert attribute="footer"/>

</body>

</html:html>
```

Utilizzando i tag <tiles:insert> e <tiles:getAsString> abbiamo semplicemente definito i punti in cui sarà inserito il contenuto della pagina.

In particolare, il tag <tiles:insert> indica a Tiles di inserire un tile, una mattonella, identificata dall'attributo attribute in un punto specifico del layout; il tag <tiles:getAsString>, invece, consente di inserire nella pagina un contenuto convertendolo prima in una stringa di testo e costituisce, in un certo senso, un modo per effettuare un passaggio di parametri.

A questo punto bisogna definire il contenuto da inserire nel template. Per far ciò Tiles utilizza le page definition, che consentono di definire tutti gli attributi che andranno a costruire una pagina. Abbiamo scelto di raccogliere le definizioni in modo centralizzato in un file XML, che l'applicazione caricherà allo startup. Per utilizzare questa funzionalità occorre dichiarare, nel file struts-config.xml, l'utilizzo del PlugIn org.apache.struts.tiles.TilesPlugin:

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
<set-property property="moduleAware" value="true"/>
<set-property property="definitions-parser-validate"
      value="true"/>
<set-property property="definitions-config" value="/WEB-
      INF/tileDefinitions.xml"/>
</plug-in>
```

Come si può vedere abbiamo impostato il PlugIn per la ricerca delle definizioni nel file tileDefinitions.xml.

Il file tileDefinitions.xml conterrà le definizioni relative a tutte le pagine utilizzate nell'applicazione. Le definizioni avranno il seguente aspetto:

```
<definition name="page.home">
      <put name="title" value="Microarray Web - Home page" />
      <put name="header" value="/pages/header.jsp" />
      <put name="menu" value="/pages/menu.jsp" />
      <put name="body" value="/pages/homepage.jsp" />
      <put name="footer" value="/pages/footer.jsp" />
</definition>
```

Con la sintassi che precede abbiamo specificato quali pagine JSP saranno utilizzate per costruire a runtime la definition di nome page.home. Ora occorrerà modificare la configurazione di ogni Action class come segue, affinchè questa effettui il forward verso una definition di Tiles, invece che verso una comune pagina JSP:

```
<action name="startForm" path="/start"
type="microarray.actions.StartAction" validate="false">
      <forward name="home" path="page.home"/>
</action>
```

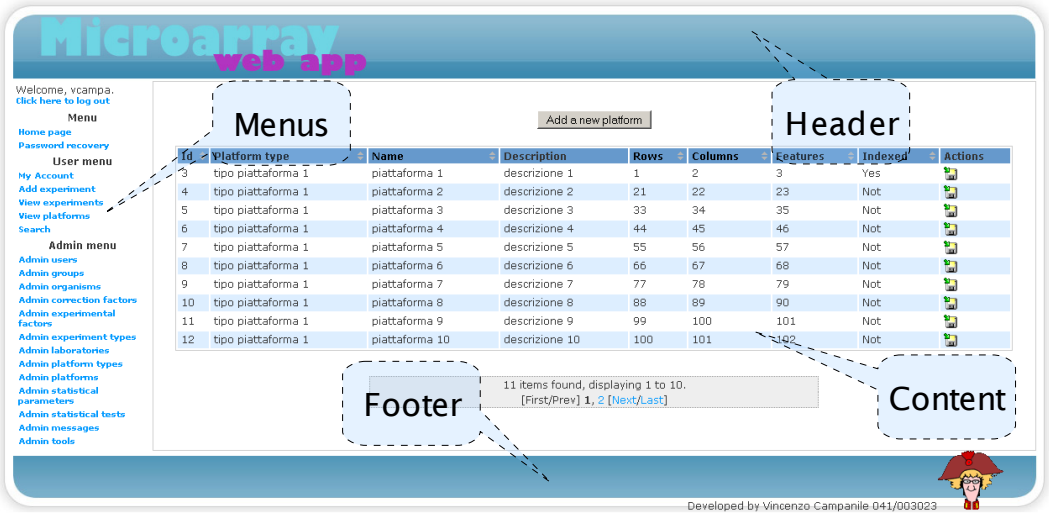

Figura 5.1.: Layout delle pagine dell'applicazione.

L'utilizzo di Tiles ci ha consentito di separare i contenuti dal layout, e quindi dalla struttura delle pagine. I fogli di stile CSS, invece, ci consentono di separare i contenuti dalla formattazione ad essi applicata. CSS sta per Cascading Style Sheets, cioè fogli di stile a cascata, una tecnologia promossa dal W3C, il World Wide Web Consortium, a partire dal 1996 che consente di fissare gli stili (per es. tipo di carattere, colori e spaziature) da applicare ai documenti HTML e XHTML.

Il punto di partenza della tecnologia CSS consiste nel fatto che il contenuto delle pagine dovesse essere sempre definito tramite codice (X)HTML, mentre la formattazione sarebbe stata trasferita altrove, in un codice completamente separato, il CSS appunto. Il punto di incontro tra i contenuti e la formattazione, tra il codice (X)HTML e i fogli di stile CSS, è negli attributi: class e id.

Per associare un foglio di stile CSS ad una pagina HTML basta utilizzare il seguente tag:

```
<head>
<link rel="stylesheet" type="text/css"
      href="../css/template_css.css" />
</head>
```

D'ora in poi il browser cercherà la formattazione da applicare agli elementi della pagina HTML (anche) nel foglio di stile template_css.css. Il linguaggio utilizzato per definire gli stili è semplice e intuitivo, ad esempio, volendo che tutti gli elementi html caratterizzati dal valore “content” dell'attributo class abbiano lo stesso aspetto, basta codificare:

```
.content {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 10px;
	color: #000000;
	font-weight: normal;
}
```

A questo punto, per definire lo stile del contenuto di una tabella, basterà associare la classe content al tag corrispondente.

```
<table class=”content”>
...
</table>
```

In verità, il linguaggio CSS consente di attribuire uno stile a singoli elementi caratterizzati dall'attributo id, a insiemi di elementi caratterizzati dal medesimo valore dell'attributo class, o anche a tutti i tag di un certo tipo. A seguire riportiamo porzioni di codice utilizzate per sfruttare le funzionalità elencate nel foglio di stile utilizzato dalla nostra applicazione.

```
// Define tag style
body {
	margin: 0px;
}

// Define class style
.left {
	width: 160;
	vertical-align: top;
	height: 20;
}

// Define id style
#active_menu {
	color: #0066CC;
}
```

5.15 DisplayTag library

Le librerie di tag JSP, o semplicemente tag library, forniscono insiemi di tag da utilizzare per la creazione di pagine JSP. L'obiettivo di tali librerie è la riduzione o anche la totale eliminazione della presenza di codice Java all'interno delle pagine JSP, in modo da consentirne la scrittura anche a chi ha poca familiarità con i linguaggi di programmazione, ma è abituato ad utilizzare linguaggi di markup come HTML o XML. Da un punto di vista organizzativo, l'utilizzo delle tag library in luogo del linguaggio di programmazione consente di assegnare la realizzazione della view ai professionisti della grafica e del design, piuttosto che a generici programmatori Java.

L'utilizzo delle tag library permette di strutturare in maniera chiara l'applicazione Web, separando, anche dal punto di vista della tecnica di implementazione, le pagine JSP di cui è composta la view, da model e controller. Infatti la view sarà creata in modo dichiarativo utilizzando i tag JSP, mentre model e controller saranno realizzati utilizzando il linguaggio di programmazione Java.

La release di Struts di cui abbiamo fatto uso porta con se quattro librerie di tag contenenti quanto è necessario alla realizzazione della view nella maggior parte delle moderne applicazioni web. Riportiamo in tabella una panoramica delle funzionalità che esse offrono.

Nome	Descrizione
struts-bean	Questa tag library contiene tag utili nell'accesso ai JavaBean e alle loro proprietà, così come alla definizione di nuovi JavaBean. Fornisce, tra gli altri, meccanismi per la creazione di nuovi bean basati sul valore di cookies, headers e parametri della request.
struts-html	Questa tag library contiene tag utili alla creazione di form per l'input di dati, così come tag utili alla creazione di user interface basate su HTML.
struts-logic	Questa tag library contiene tag utili nella creazione condizionale dell'output, nell'iterazione sulle collezioni di oggetti e nella gestione del flusso applicativo.
struts-nested	Questa tag library estende agli oggetti innestati le funzionalità offerte dalle librerie precedenti. I JavaBean innestati sono bean che includono al loro interno un riferimento ad un ulteriore bean, in modo che l'accesso a quel bean debba necessariamente passare per il bean di livello superiore.

Le tag library elencate sono generaliste e consentono di adempiere alla maggior parte dei compiti. Tuttavia può essere necessario servirsi di librerie più specializzate per semplificare ulteriormente la creazione delle pagine. In una applicazione data centrica come quella che abbiamo realizzato, ad esempio, è frequente la necessità di stampare sotto forma tabellare le proprietà di collezioni di oggetti, magari spalmando le tabelle su più pagine.

L'iterazione su collezioni di oggetti e la stampa delle loro proprietà in forma tabellare può avvenire combinando opportunamente i tag delle librerie logic, nested e bean di Struts. Purtroppo, però, questa soluzione non è scalabile: al complicarsi dei dati da stampare aumenta presto la complessità e conseguentemente diminuisce la leggibilità e manutenibilità del codice.

Ci viene in aiuto DisplayTag library, una libreria di tag open source specializzata nella stampa e impaginazione di tabelle. L'obiettivo degli autori di questa libreria è fornire dei tag che possano essere utilizzati per la parte view di una qualunque applicazione web realizzata secondo il modello MVC (Model-View-Controller). L'utilizzo dei tag di DisplayTag consente allo sviluppatore di concentrarsi sul

reperimento e l'organizzazione dei dati, delegando alla libreria le problematiche relative alla loro visualizzazione in forma tabellare.

Le funzionalità principali che DisplayTag library offre sono le seguenti:

- definizione automatica delle colonne di una tabella sulla base delle proprietà degli oggetti che la tabella deve visualizzare;
- paging o paginazione, cioè divisione su più pagine, del contenuto delle tabelle;
- sorting, l'ordinamento automatico delle righe di una tabella sulla base dei valori contenuti in una colonna;
- personalizzazione dell'aspetto delle tabelle mediante fogli di stile CSS;
- formattazione e post processing dei valori contenuti in ogni colonna per mezzo di classi Decorator;
- esportazione della tabella nei più comuni formati (CSV, Excel, XML e PDF);
- supporto all'internazionalizzazione dei testi visualizzati mediante resource bundle.

A queste si aggiunge la possibilità di gestire esternamente il paging e il sorting integrando la logica applicativa con l'utilizzo della libreria.

Per poter utilizzare la DisplayTag non dobbiamo far altro che salvare la collezione di oggetti da stampare in un attributo della request. Ciò può essere fatto all'interno di una qualsiasi delle nostre action con una istruzione del tipo:

```
request.setAttribute(“nomeLista”,list);
```

Dove l'oggetto list può essere un oggetto di una qualiasi delle seguenti tipologie:

- una Collection,
- un Enumeration,
- una Map (i valori sono stampati per righe),
- un Dictionary (i valori sono stampati per righe),
- un array,
- un Iterator.

Fatto questo, stamperemo una tabella nella nostra pagina JSP, semplicemente con la sintassi:

```
<display:table name="nomeLista"/>
```

In questo modo lasciamo a DisplayTag l'incombenza di individuare tutti gli attributi degli oggetti raccolti da List e di tabellarli costruendo una colonna per ogni attributo.

Per ottenere la paginazione automatica delle tabelle troppo lunghe è sufficiente specificare un valore per l'attributo pagesize, ad esempio:

```
<display:table name="nomeLista" pagesize="10"/>
```

e sarà DisplayTag a dividere la tabella su più pagine e a gestire la navigazione tra le pagine ottenute.

Se invece volessimo stampare solo alcune proprietà degli oggetti nella collezione, allora potremmo innestare, nel tag <display:table> il tag <display:column>, come segue:

```
<display:table name="test">
      <display:column property="property1"/>
      <display:column property="property2"/>
      <display:column property="property3"/>
      <display:column property="property4"/>
      <display:column property="property5"/>
</display:table>
```

Possiamo sfruttare le funzionalità di internazionalizzazione della libreria includendo nel tag <display:column> l'attributo titleKey, mediante il quale chiederemo a DisplayTag di cercare il titolo della colonna nel resource bundle.

```
<display:column property="username"
      titleKey="userstable.username"/>
```

Per ottenere l'ordinamento del contenuto di una colonna basta utilizzare l'attributo sortable dello stesso tag, come segue:

```
<display:column property="username"
      titleKey="userstable.username" sortable=”true”/>
```

e DisplayTag introdurrà nella colonna username della tabella stampata un pulsante per l'ordinamento ascendente/discendente e gestirà questa funzionalità in modo trasparente allo sviluppatore.

5.15.1 External paging & sorting

Abbiamo visto come DisplayTag applica le funzionalità di paging e sorting, in modo completamente trasparente allo sviluppatore, alla collezione di oggetti memorizzata nel requestScope. Un problema potrebbe essere costituito dalla necessità di memorizzare nella request un oggetto di enormi dimensioni, quale può essere una collezione dei dati prelevati da una tabella di centinaia di migliaia di righe.

Supponiamo, ad esempio (e per assurdo!), che la nostra applicazione, dopo qualche anno di utilizzo, si trovi a gestire 10000 utenti registrati. Se volessimo stampare una tabella contenente i dati di tutti gli utenti registrati, come facciamo in realtà, sfruttando il paging automatico, sarebbe necessario salvare nella request un ArrayList costituita da 10000 istanze diverse dell'oggetto UserDTO, utilizzato per raccogliere i dati dalla tabella users. All'amministratore che accede a questa funzionalità DisplayTag mostrerebbe una tabella contenente solo i dati di qualche decina dei 10000 utenti raccolti dal database: ecco un significativo spreco di risorse.

DisplayTag ci ha consentito di risolvere questo problema gestendo esternamente il paging e il sorting. Abbiamo modificato, allo scopo, tutti i metodi che si occupano della lettura dei dati presenti nel database per generare liste, limitando le query esattamente al numero di valori che ogni pagina deve mostrare. Omettiamo le complicazioni introdotte dall'utilizzo di questa tecnica per esporre la principale conseguenza dell'utilizzo di questa tecnica: tutte le query SQL del tipo:

```
SELECT * FROM nometabella;
```

sono state trasformate in query del tipo:

```
SELECT * FROM nometabella
      ORDER by nomecolonna ASC/DESC
      LIMIT inizio, numerorecord;
```

Ottenendo così un utilizzo più razionale del DBMS, oltre che un risparmio di risorse.

5.15.2 I TableDecorator

DisplayTag già include una funzionalità per la decorazione del contenuto di una tabella detta smartlinking, che consiste nella trasformazione automatica di un url o un indirizzo di posta contenuto in una tabella, in un link cliccabile. Ad esempio, nella stampa di una tabella contenente i dati personali degli utenti, lo smartlinking consente di trasformare la semplice variabile testuale che costituisce l'indirizzo e-mail in un tag HTML per l'invio di un messaggio.

L'utilizzo delle classi Decorator consente di estendere questo tipo di funzionalità alla decorazione di ogni attributo. Ad esempio, nello sviluppo dell'applicazione, ci siamo posti il problema di internazionalizzare il valore dell'attributo gender della tabella users, memorizzato come tinyint(1) nel database. In particolare, abbiamo utilizzato la convenzione di rappresentare con il valore 0 il sesso maschile, e con il valore 1 il sesso femminile.

Una interfaccia di amministrazione che mostri una lista degli utenti contenente i valori 0 o 1 nella colonna gender sarebbe decisamente poco user friendly. Avremmo preferito decisamente che la colonna contenesse i valori male e female, internazionalizzati mediante resource bundle. Per ottenere ciò abbiamo operato come segue.

Abbiamo creato i seguenti messaggi nei nostri file properties:

```
userstable.gender.0=Male
userstable.gender.1=Female
```

nel file in lingua inglese, e

```
userstable.gender.0=Maschio
userstable.gender.1=Femmina
```

nel file in lingua italiana. Poi abbiamo creato una classe UsersListDecorator che estende la classe org.displaytag.decorator.TableDecorator. Abbiamo introdotto in questa classe un metodo getGender() che effettua una sorta di overriding dello stesso metodo dell'oggetto contenitore UserDTO, come segue:

```
public class UsersListDecorator extends TableDecorator {

private MessageResources props;

public void init(javax.servlet.jsp.PageContext context,
      java.lang.Object decorated) {
      props = (MessageResources)context.getRequest().
            getAttribute(Globals.MESSAGES_KEY);
}

public UsersListDecorator() {
      super();
}

public String getGender() {
      int storedvalue =
      ((UserDTO)this.getCurrentRowObject()).getGender();
      String key = new String();
      key = "userstable.gender." + storedvalue;
      return props.getMessage(key);
}

}
```

Come si può vedere questo TableDecorator utilizza un metodo init() di inizializzazione, utilizzato, in questo caso, per ottenere un riferimento al MessageResources di Struts. Il metodo getGender() recupera il valore intero con cui è memorizzato il sesso dell'utente in ogni riga della tabella, e lo trasforma in una stringa internazionalizzandola.

Questo semplice esempio dovrebbe essere sufficiente ad intuire le potenzialità che l'oggetto TableDecorator può esprimere. L'utilizzo di questo oggetto ci ha consentito di ottenere, con la stessa semplicità, l'internazionalizzazione di una data presente nel database o l'impaginazione dei risultati della ricerca in forma tabellare, sempre separando nettamente la view dal model.

5.16 L'applet VIMIDA

VIMIDA è una applet Java per la visualizzazione bidimensionale dei dati di espressione genica per mezzo della principal component analysis (PCA), lineare e non lineare, e dell'algoritmo Elastic Map. E' distribuita gratuitamente dal servizio bioinformatico dell'Institut Curie di Parigi, ma non è un prodotto open source. Come semplici utilizzatori di questo prodotto ci siamo preoccupati esclusivamente di renderlo interoperabile con il nostro sistema.

Vimida ha bisogno che i dati, nel linguaggio del nostro sistema dati normalizzati o liste di geni regolati, siano espressi in un particolare formato che prevede quanto segue. I dati devono essere riportati in tabelle, delimitate dal carattere di tabulazione, dallo spazio o da una virgola, nel formato di un file di testo. La tabella non deve contenere i nomi delle colonne nella prima riga, ma il file deve includere un header che definisce i tipi e le descrizioni delle colonne. Il formato dell'header è il seguente:

```
<number_of_columns> <number_of_rows>
<Field1_name> <Field1_type> [<Field1_description>]
<Field2_name> <Field2_type> [<Field2_description>]
...
<Fieldn_name> <Fieldn_type> [<Fieldn_description>]
```

L'applet accede ai dati per mezzo di un URL, per questo motivo è necessario che siano salvati in un file temporaneo esposto sul web per mezzo di un web server. Così, ogni volta che l'utente chiede di visualizzare dei dati per mezzo di VIMIDA, il sistema provvederà a prelevare i dati dal database, a decomprimerli, a convertirli nel formato supportato dall'applet e infine e a salvarli in un file temporaneo in una sotto-cartella della deploy folder dell'applicazione.

Per evitare la proliferazione dei file temporanei sul server, abbiamo scelto di salvare i dati in un file con nome lo username dell'u-

tente che richiede la visualizzazione. In questo modo sul server ci sarà al più un file temporaneo per ogni utente registrato, invece che un file per ogni esperimento o analisi. Questo file sarà sovrascritto ogni volta con i dati dell'esperimento o analisi che l'utente vorrà visualizzare. Nel paragrafo seguente ci porremo il problema dell'eliminazione di questi file.

5.16.1 Listener per la cancellazione dei file temporanei

Abbiamo scelto di rimuovere il file temporaneo associato ad ogni utente, semplicemente allo scadere della sessione. Per ottenere questa funzionalità abbiamo creato una semplice implementazione dell'interfaccia javax.servlet.http.HttpSessionListener. L'interfaccia prevede due soli metodi:

- il metodo sessionCreated(HttpSessionEvent event), che viene invocato dal servlet container ogni volta che una sessione viene creata.
- il metodo sessionDestroyed(HttpSessionEvent event), che viene invocato dal servlet container ogni volta che una sessione viene distrutta.

Implementando questi metodi è possibile arricchire una applicazione web di funzionalità correlate agli eventi di creazione e distruzione delle sessioni come, ad esempio, un contatatore degli utenti online. Noi, invece che implementare una funzionalità del genere, abbiamo sfruttato il metodo sessionDestroyed per rimuovere il file temporaneo associato ad ogni utente che abbia utilizzato VIMIDA, operando come segue.

```
public class MySessionListener implements HttpSessionListener
{

public void sessionCreated(HttpSessionEvent event) {

}
```

```
public void sessionDestroyed(HttpSessionEvent event) {
      HttpSession session = event.getSession();
      UserDTO user = (UserDTO)session.getAttribute("user");
      if (user != null) {
            String userName = user.getUsername();
            String tempPath = session.getServletContext()
                  .getRealPath("temp");
            File file = new File(tempPath + File.separator +
                  userName + ".dat");
            file.delete();
      }
}

}
```

Poiché il compito di creare e distruggere le sessioni è riservato al servlet container, è necessario configurare il nostro Listener nel deployment descriptor, il file web.xml, dell'applicazione.

```
<listener>
<listener-class>microarray.utils.
      MySessionListener</listener-class>
</listener>
```

5.17 Luke, Lucene Index Toolbox

Luke è un tool open source utile allo sviluppo, nonché alla diagnosi dei problemi, di indici realizzati con Lucene che consente di monitorarne e modificarne il contenuto. Alcune delle funzionalità supportate da Luke sono le seguenti:

- browsing per documento o per termine;
- visualizzazione dei Document indicizzati;
- costruzione di liste dei termini più frequenti;
- eseguire ricerche e visualizzarne i risultati;
- analizzare i risultati della ricerca;
- eliminazione selettiva dei Document;
- ricostruzione dei Field originali, modifica e reindicizzazione;
- ottimizzazione e unlock forzato degli indici.

Questo tool mette a disposizione dello sviluppatore, tramite un interfaccia semplice ma ricca di funzionalità, la possibilità di operare sull'indice senza la necessità di accedere all'API di Lucene programmaticamente. Consente, inoltre, di ottenere informazioni sulle performance della ricerca, visualizzando il tempo impiegato da ogni query.

In fase di progettazione dell'indice, Luke ha sopperito alla mancanza di una interfaccia di ricerca, consentendoci di valutare il comportamento di Lucene, anche quando si erano sviluppate solo le procedure di indicizzazione. Così abbiamo ottenuto subito i feedback che altrimenti avremmo ottenuto dopo aver realizzato l'interfaccia per la ricerca, minimizzando l'insorgere di quegli effetti collaterali che poi sarebbero stati difficili da recuperare.

Secondo noi, comunque, il maggior pregio di Luke sta nell'aiuto che esso offre in fase di debug. La combinazione delle informazioni ottenute tramite il debugger di Eclipse, alle informazioni sui dati effettivamente indicizzati, ottenute tramite Luke, ha semplificato enormemente ogni intervento di manutenzione sul sottosistema di indicizzazione e ricerca.

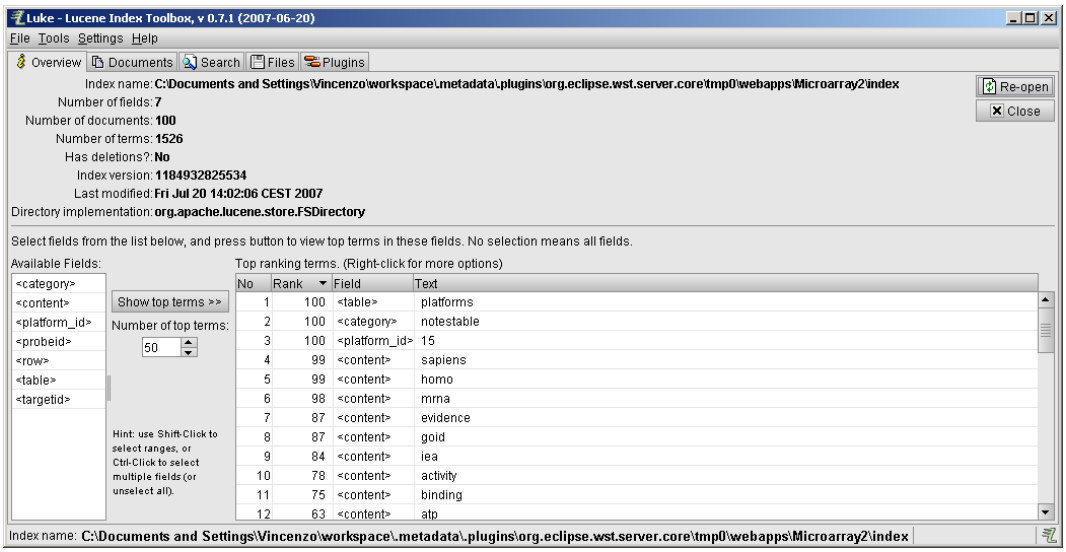

Figura 5.2.: Finestra principale di Luke, Lucene Index Toolbox.

5.18 Screen shots

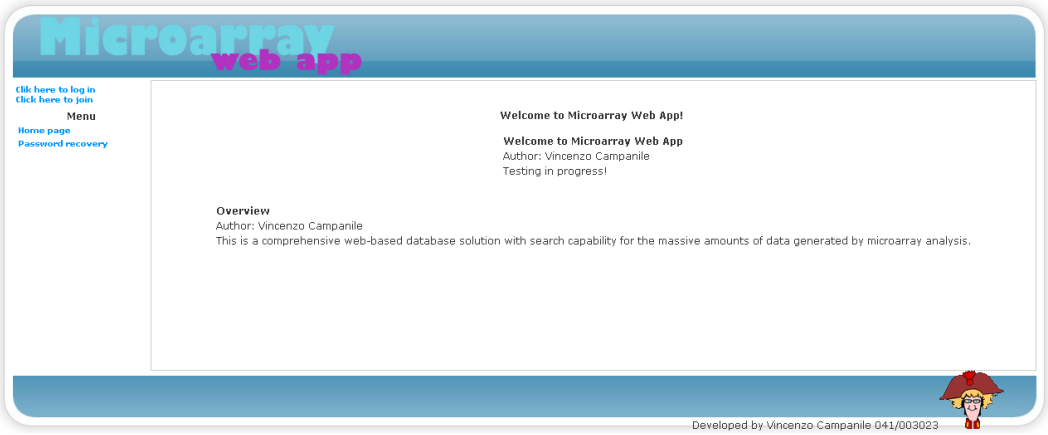

Figura 5.3.: Home page dell'applicazione vista da un visitatore anonimo. I menu sono ristretti alle sole funzionalità pubbliche.

Figura 5.4.: Il sistema notifica l'avvenuto log in di un utente. In aggiunta ai menu pubblici, è visibile il menu utente, che da accesso alle funzionalità concesse a agli utenti registrati.

Figura 5.5.: Home page dell'applicazione vista da un amministratore. Sono visibili tutti i menu, incluso quello di amministrazione.

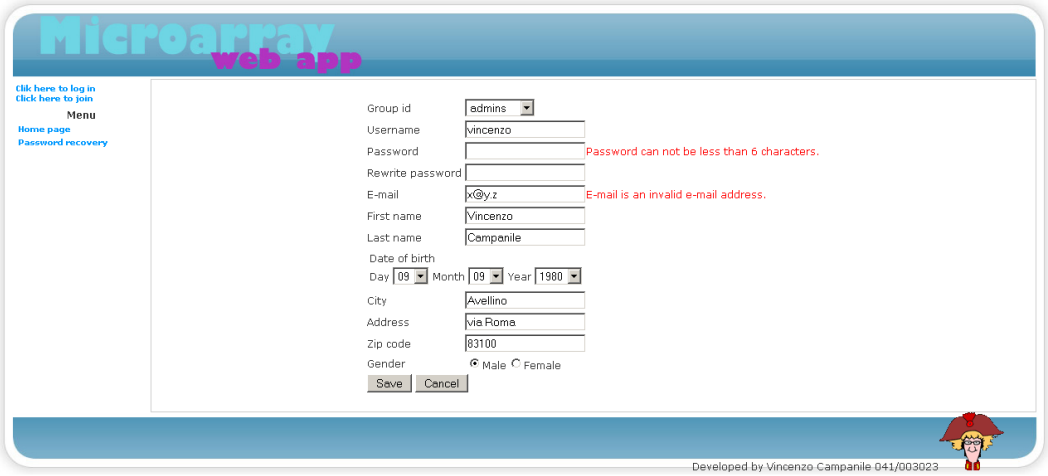

Figura 5.6.: Pagina di registrazione. L'immissione di dati in un formato non consentito ha causato la risposta di Validator con la conseguente visualizzazione dei messaggi di errore.

Figura 5.7.: Pagina di log in. La selezione della checkbox attiva la funzionalità di autenticazione automatica, in modo che nei futuri accessi non sia necessario passare per questa pagina.

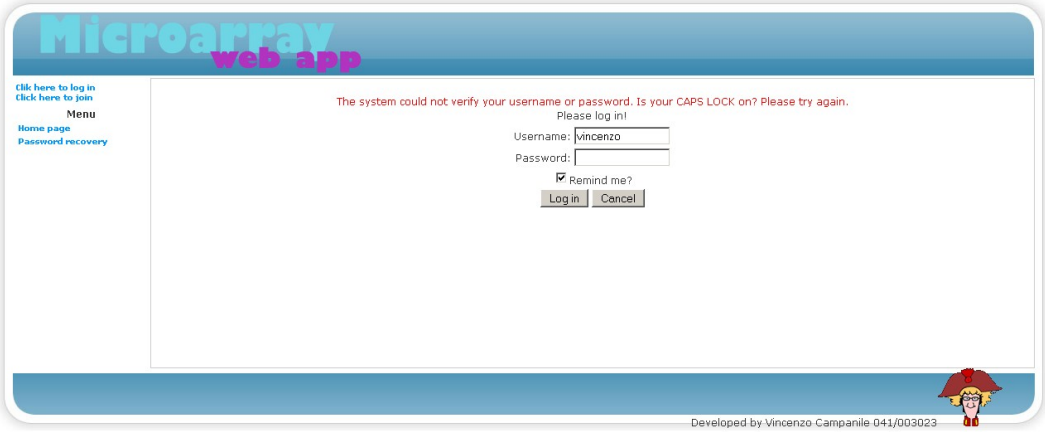

Figura 5.8.: Risultato di un tentativo di accesso con dati non corretti. Il sistema consiglia di fare attenzione nell'immissione della password.

Figura 5.9.: Interfaccia per il recupero di password smarrite. L'immissione dello username di un utente registrato causa l'invio della password, a mezzo posta elettronica, all'indirizzo di posta con cui si è registrato.

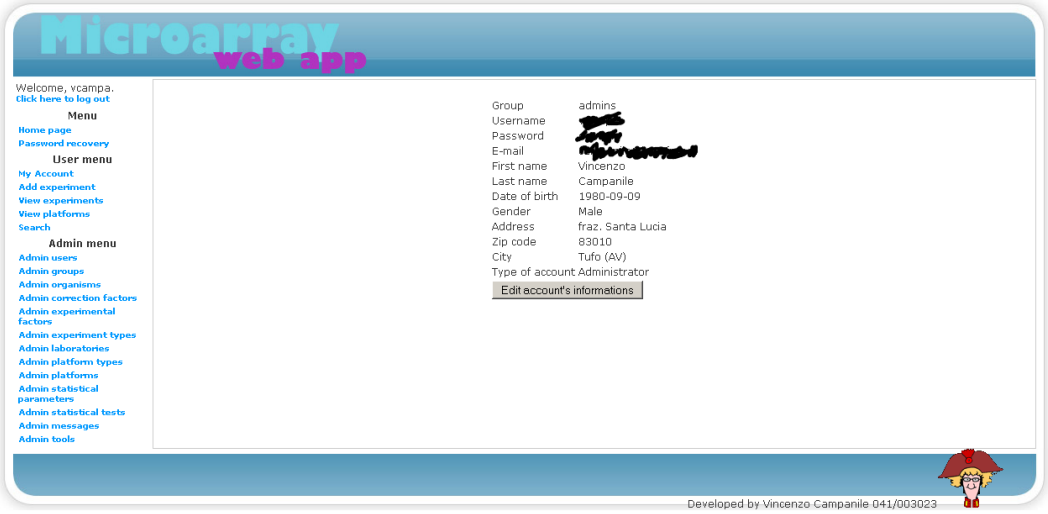

Figura 5.10.: Visualizzazione dei dettagli dell'account utente. Il pulsante da accesso alla pagina per la modifica dei propri dati personali.

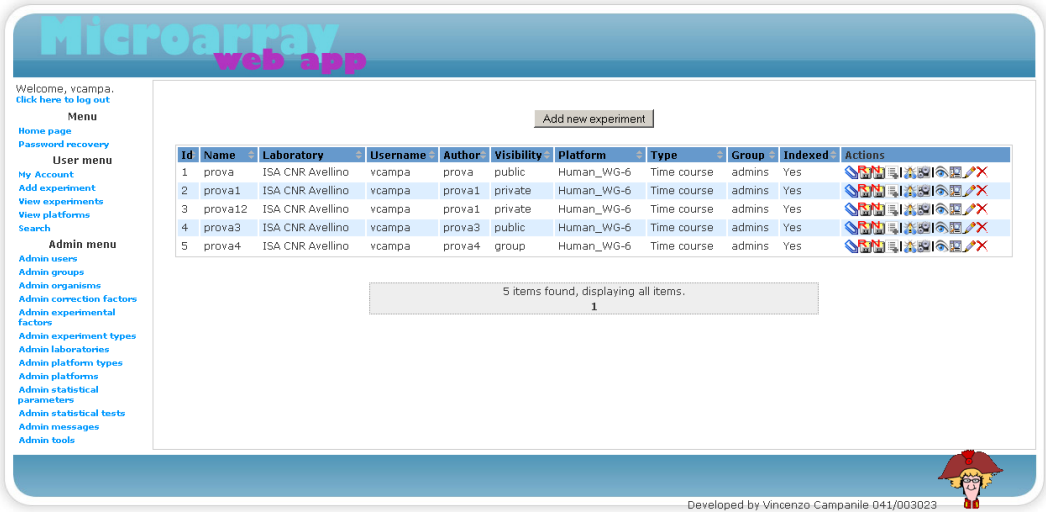

Figura 5.11.: Pagina di visualizzazione degli esperimenti archiviati. L'accesso alla pagina è consentito solo ad amministratori ed utenti registrati. I pulsanti sulla destra consentono nell'ordine: download allegato, download dati grezzi, download dati normalizzati, indicizzazione, visualizzazione campioni, visualizzazione analisi, visualizzazione dei dati normalizzati con VIMIDA, vista dettagli, modifica ed eliminazione.

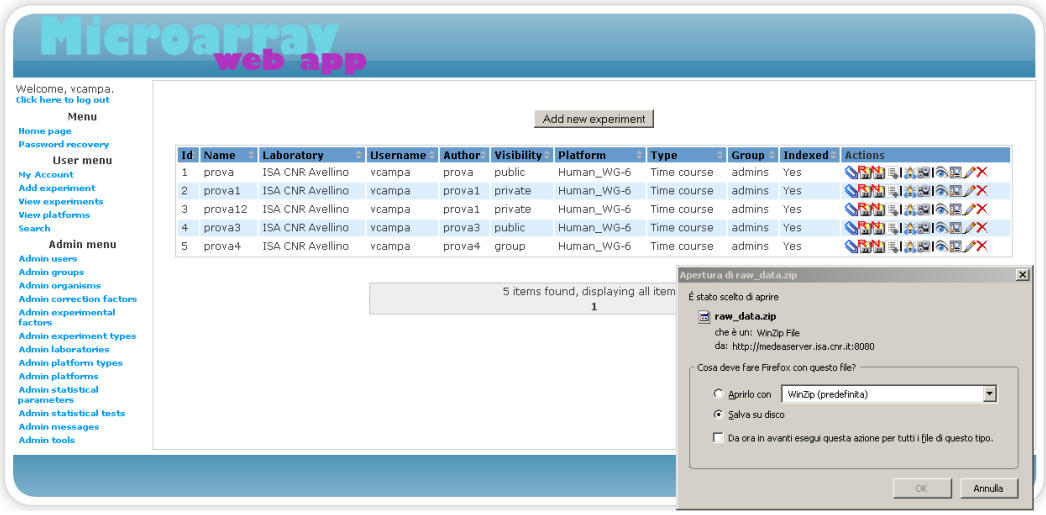

Figura 5.12.: Download dei dati grezzi. Il sistema offre al browser il file compresso estratto dal database.

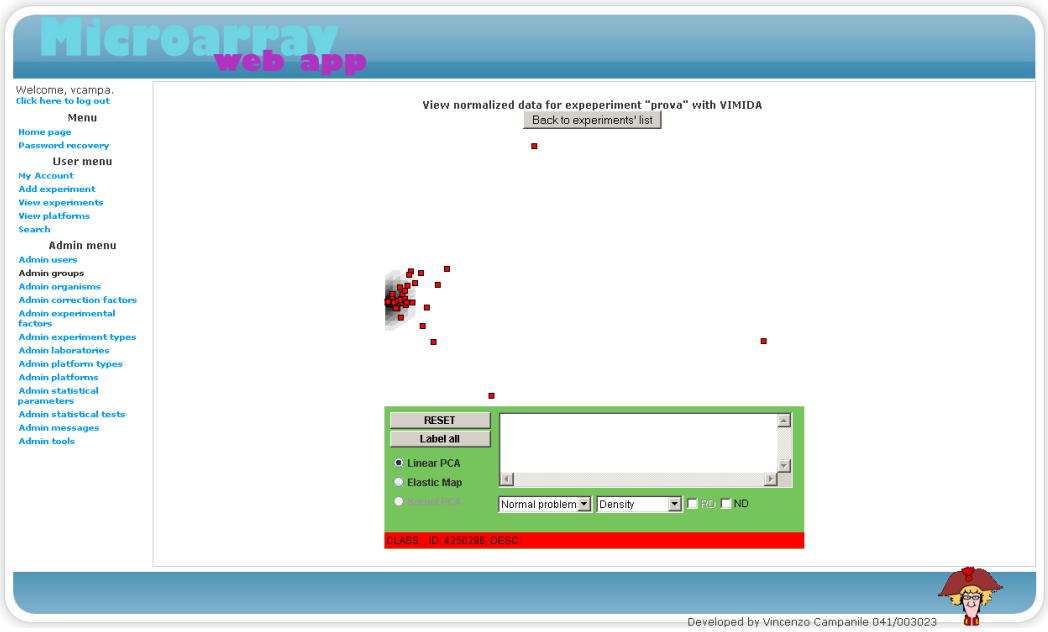

Figura 5.13.: Visualizzazione dei dati normalizzati con l'applet VIMIDA.

Figura 5.14.: Visualizzazione dei dettagli di un esperimento. I pulsanti consentono il download dei dati; i link, invece, consentono la visualizzazione dei campioni e delle analisi associate.

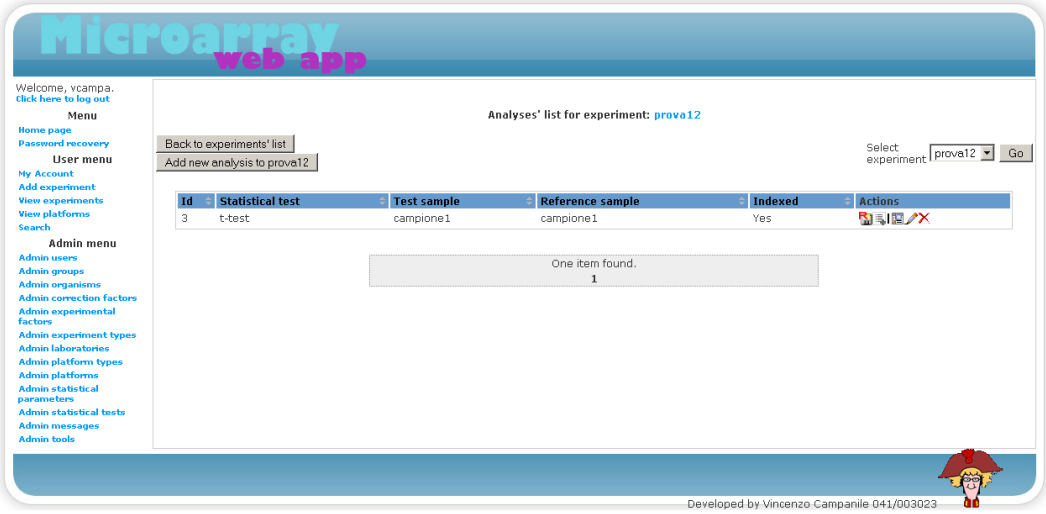

Figura 5.15.: Visualizzazione delle analisi associate ad un esperimento. La drop down list in alto a destra consente la navigazione tra gli esperimenti. I pulsanti sulla destra consentono, nell'ordine, il download dei geni regolati, l'indicizzazione, la visualizzazione dei dettagli, la modifica e l'eliminazione.

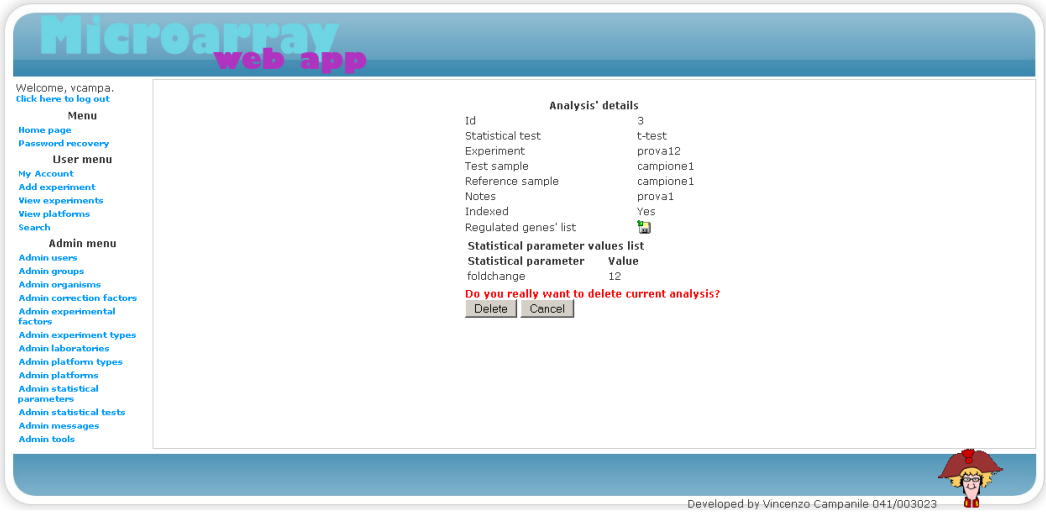

Figura 5.16.: Pagina per l'eliminazione di una analisi. La pagina include i dettagli sull'analisi per consentire una più attenta scelta dei dati da eliminare.

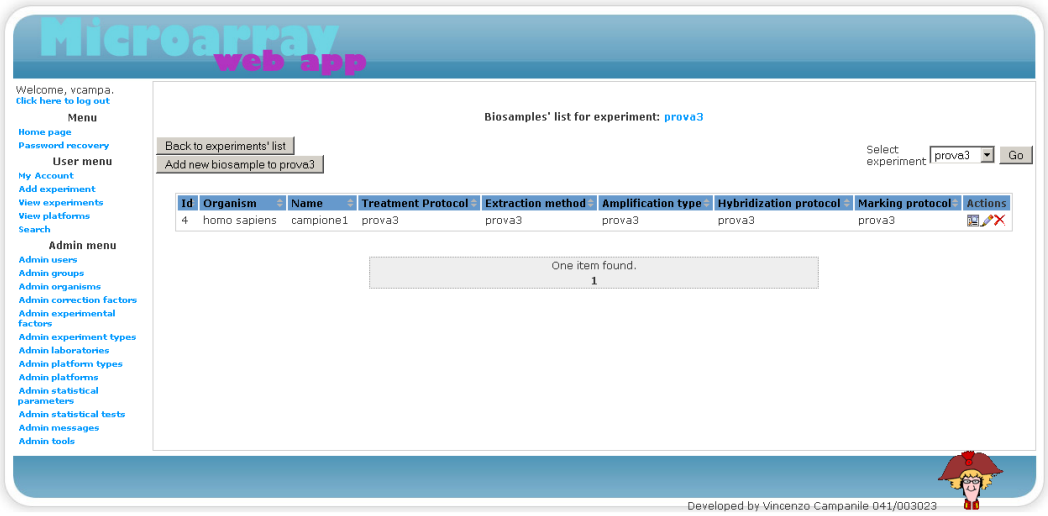

Figura 5.17.: Visualizzazione dei campioni associati ad un esperimento. La drop down list in alto a destra consente la navigazione tra gli esperimenti. I pulsanti consentono, nell'ordine, la visualizzazione dei dettagli, la modifica e l'eliminazione.

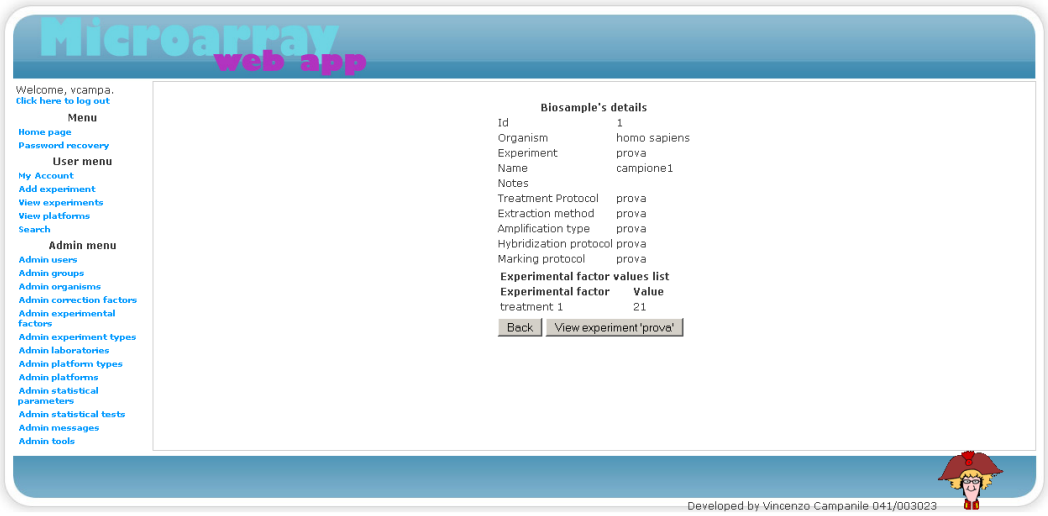

Figura 5.18.: Dettagli di un campione. In basso sono riportati i valori assunti dai fattori sperimentali utilizzati.

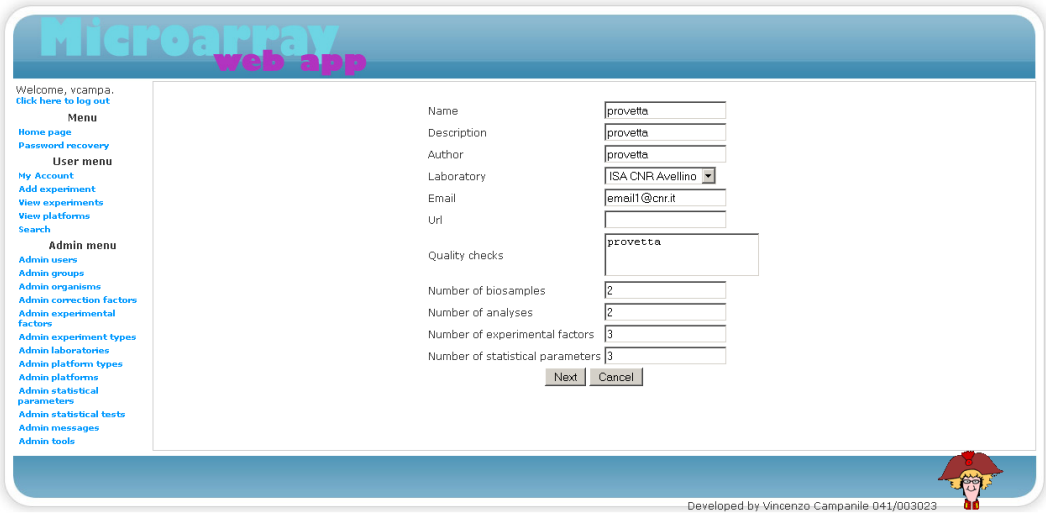

Figura 5.19.: Prima pagina del wizard per l'inserimento di un nuovo esperimento.

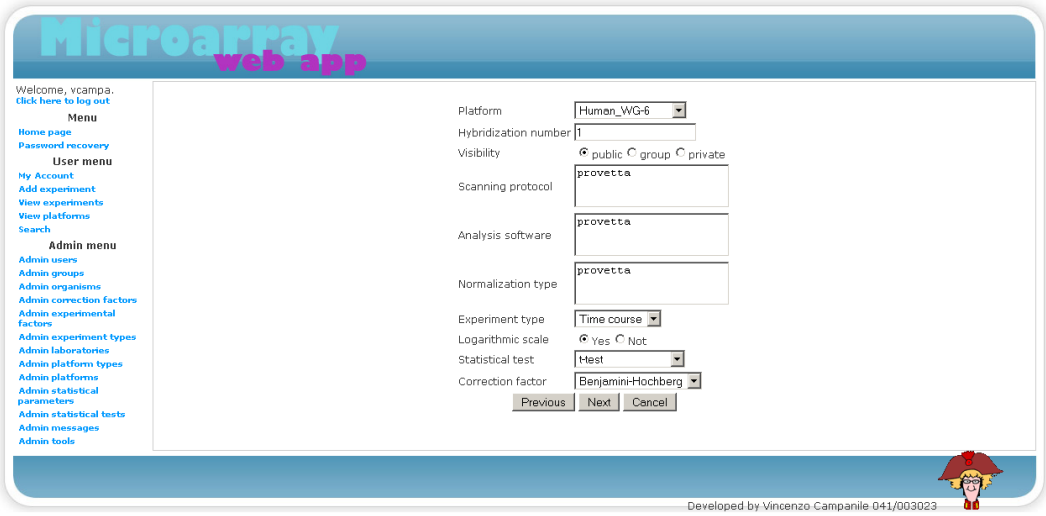

Figura 5.20.: Seconda pagina del wizard per l'inserimento di un nuovo esperimento.

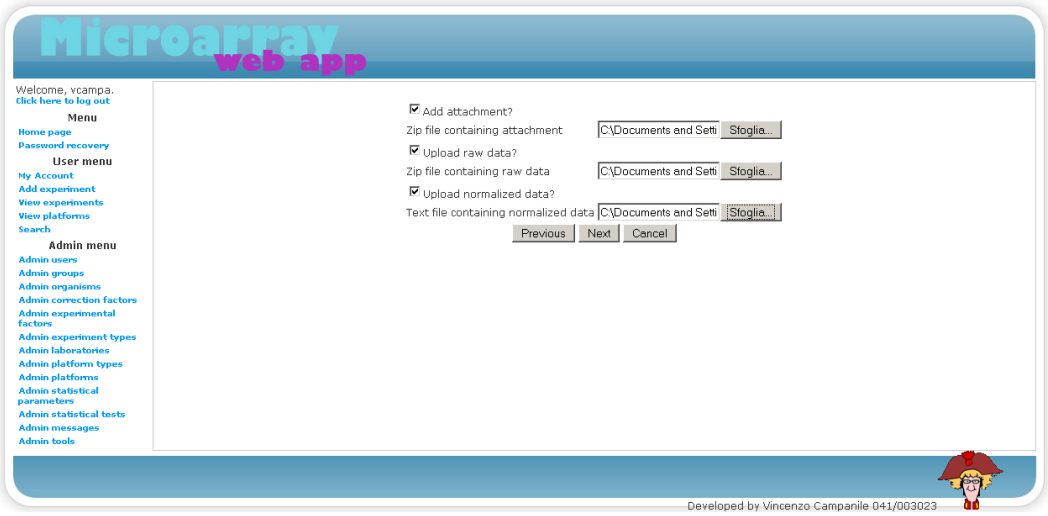

Figura 5.21.: Terza pagina del wizard per l'inserimento di un esperimento. E' possibile scegliere se caricare o meno ognuno dei dati associati all'esperimento.

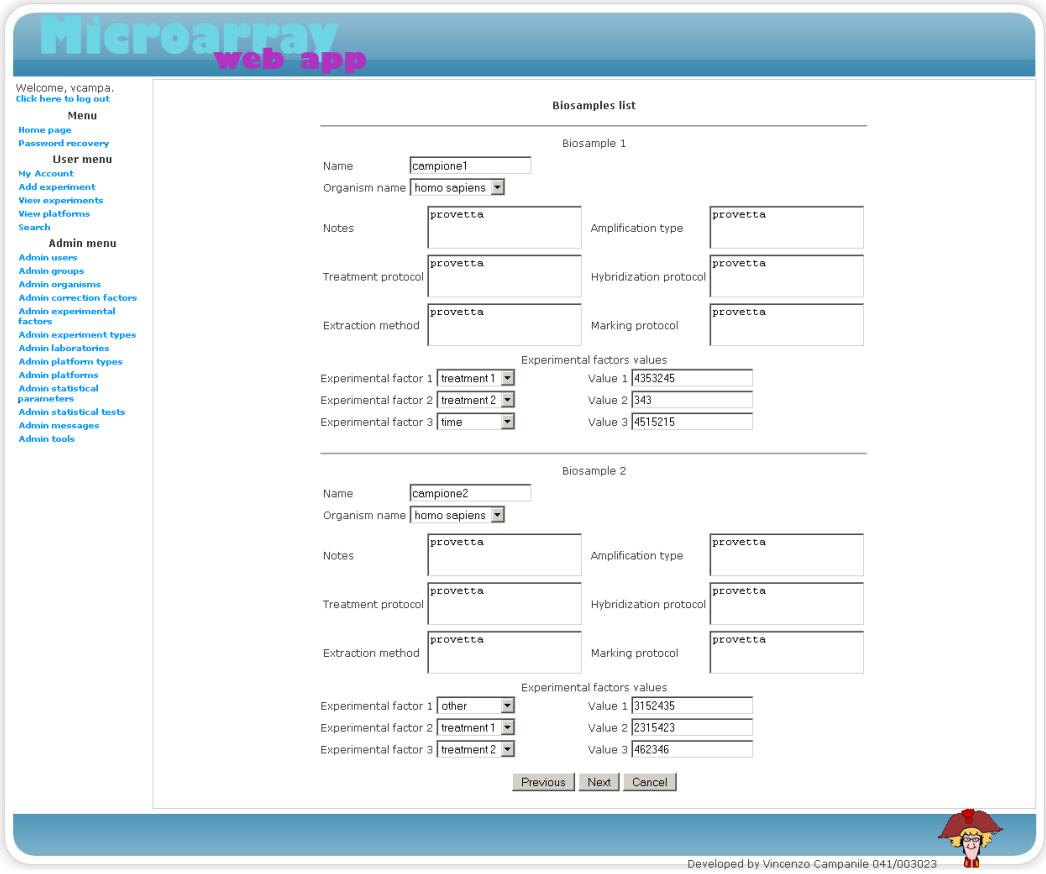

Figura 5.22.: Quarta pagina del wizard per l'inserimento di un nuovo esperimento. Consente di definire i dettagli dei campioni e dei corrispondenti fattori sperimentali, nelle quantità definite nella prima pagina del form.

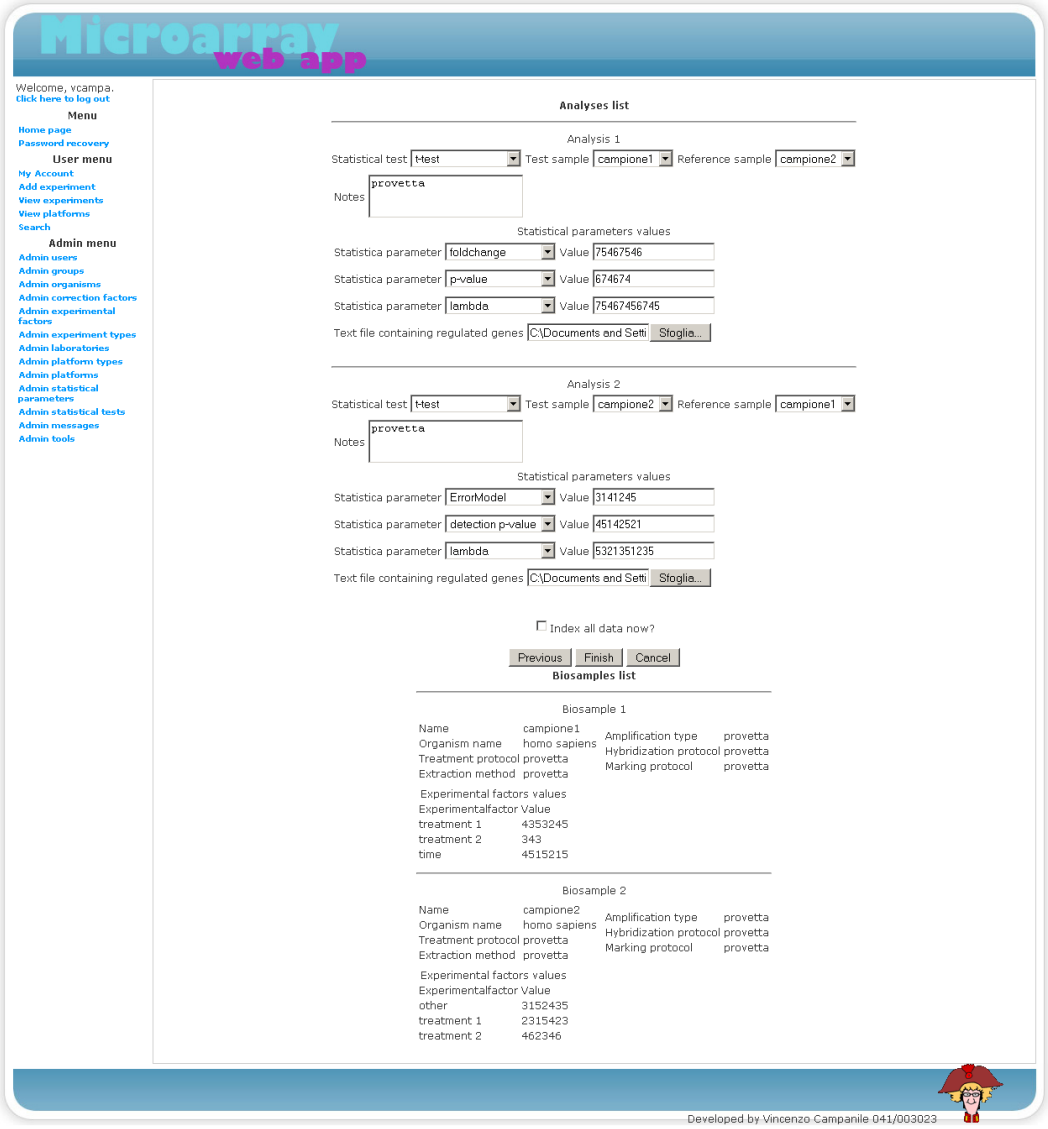

Figura 5.23.: Ultima pagina del wizard per l'inserimento di un esperimento. La pagina è destinata all'inserimento delle analisi, delle rispettive liste di geni regolati e dei parametri statistici, nelle quantità stabilite nella prima pagina del wizard. La checkbox consente di scegliere se indicizzare subito o meno i dati immessi.

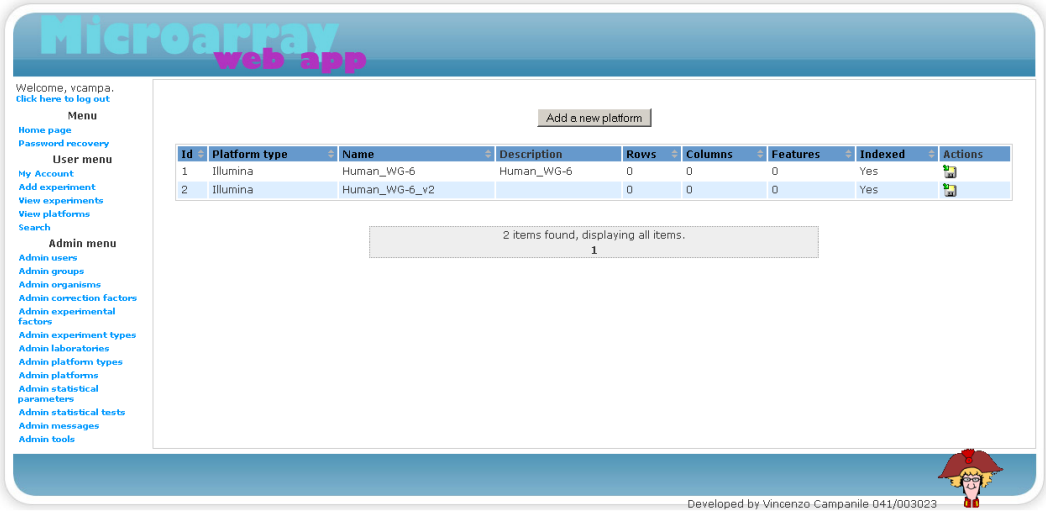

Figura 5.24.: Pagina per la visualizzazione delle piattaforme archiviate. La pagina destinata agli utenti differisce per l'assenza dei pulsanti destinati all'inserimento, eliminazione e modifica dei dati, ma include il pulsante per il download della tabella delle annotazioni.

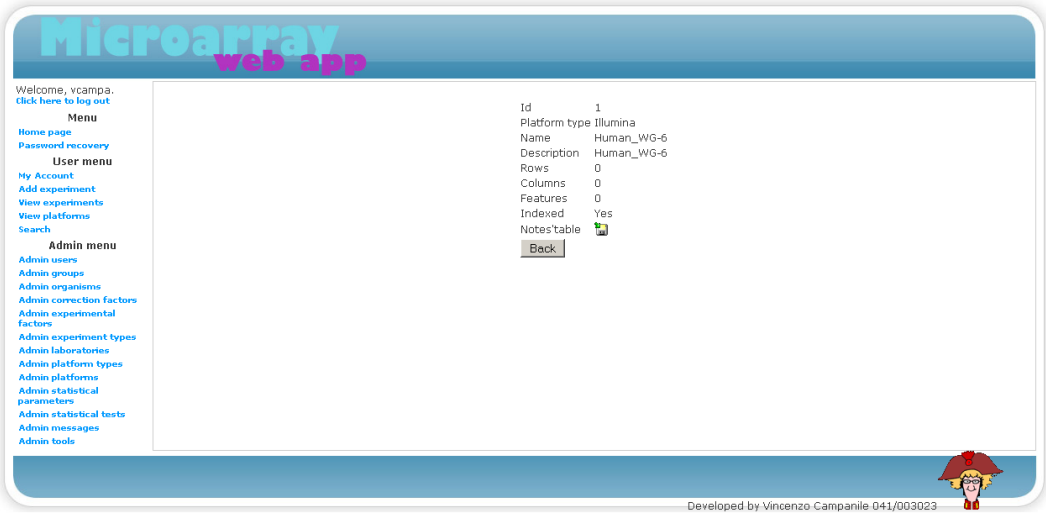

Figura 5.25.: Dettagli di una piattaforma. Il pulsante in basso consente il download della tabella delle annotazioni.

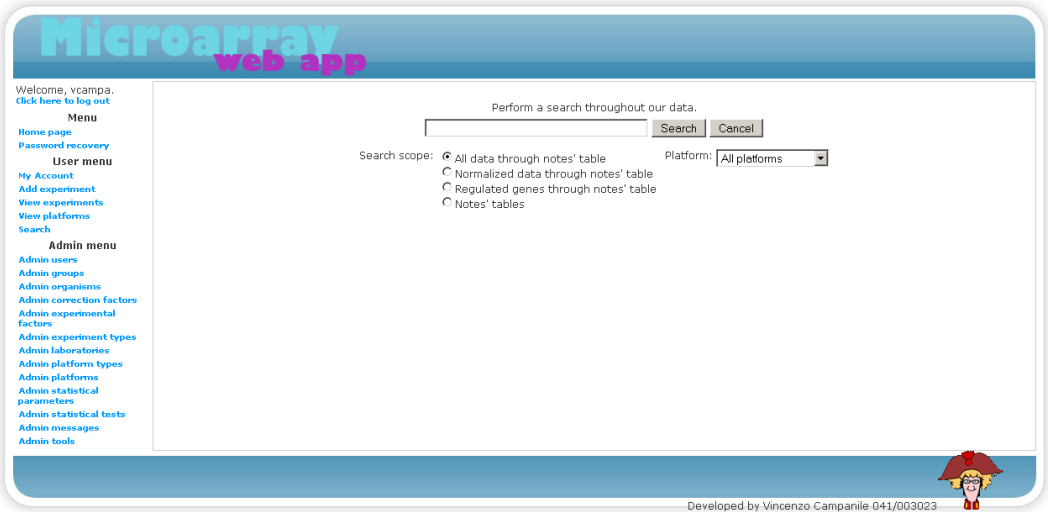

Figura 5.26.: Pagina per la ricerca. Il radio button consente di scegliere la modalità di ricerca mentre la drop down list offre la possibilità di restringere la ricerca ad una particolare piattaforma.

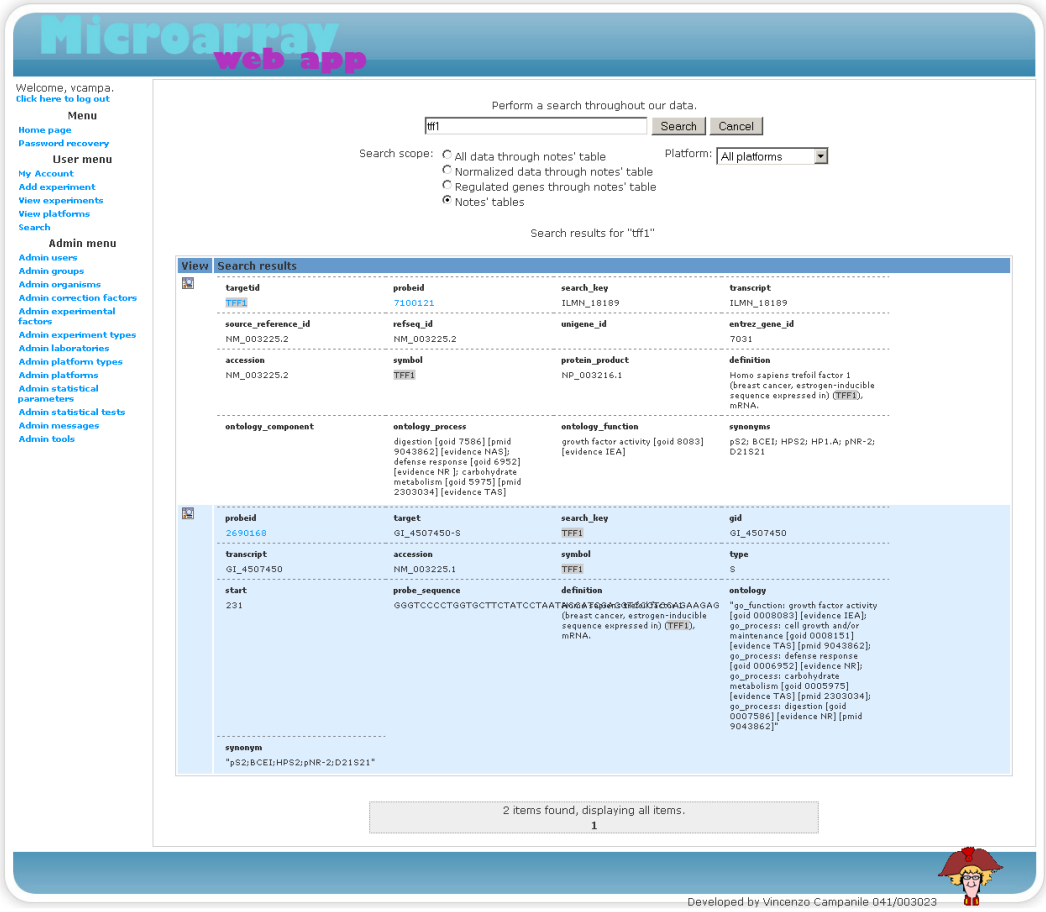

Figura 5.27.: Risultati di una ricerca effettuata sulle tabelle delle annotazioni. Il pulsante a destra di ogni risultato offre la possibilità di visualizzare i dettagli della piattaforma da cui è stato estratto il dato. Le parole trovate sono evidenziate in grigio.

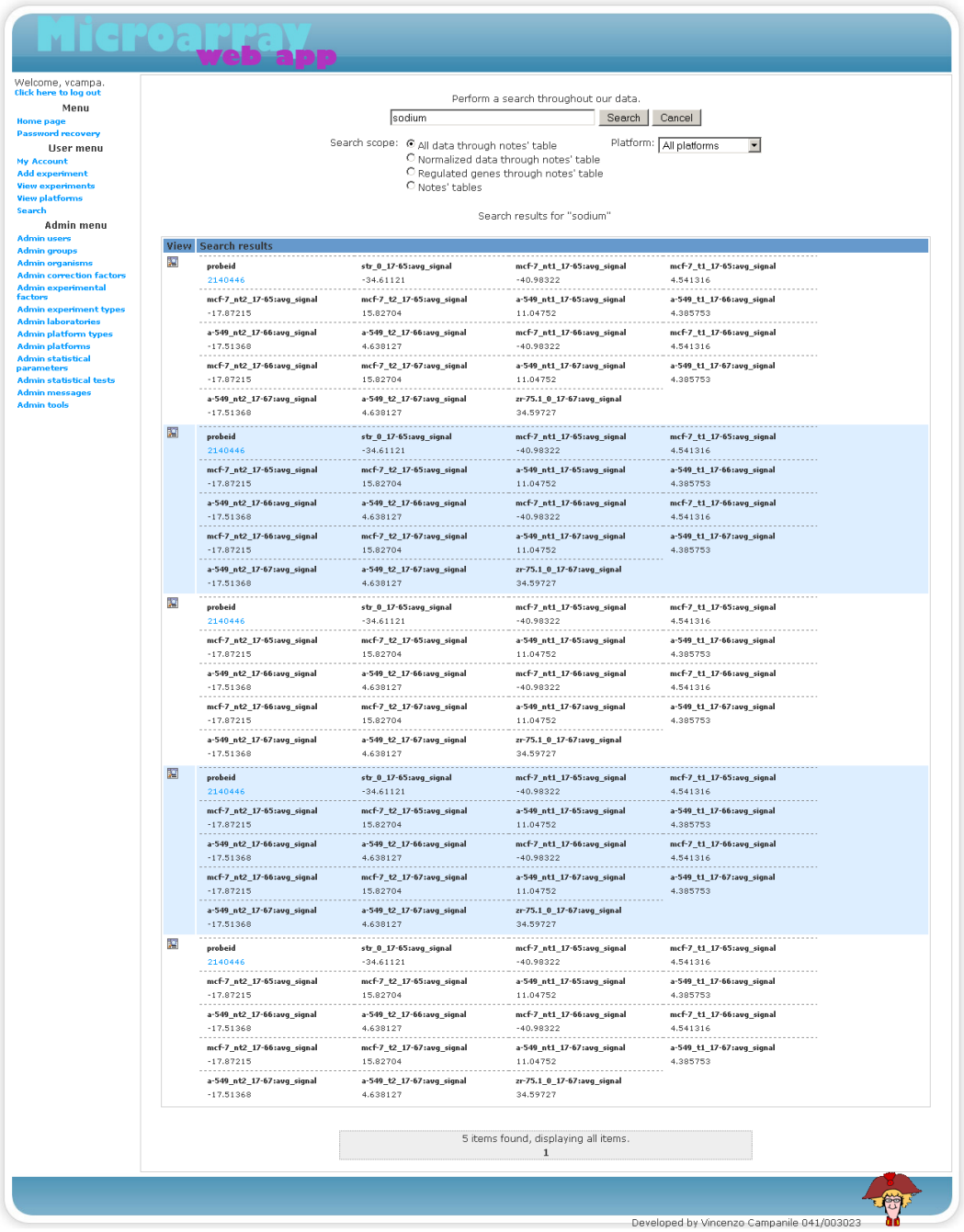

Figura 5.28.: Risultati di una ricerca automatizzata tra i dati (dati normalizzati e geni regolati). Il pulsante a destra di ogni riga consente di accedere ai dettagli relativi all'esperimento, o all'analisi, cui afferiscono i dati.

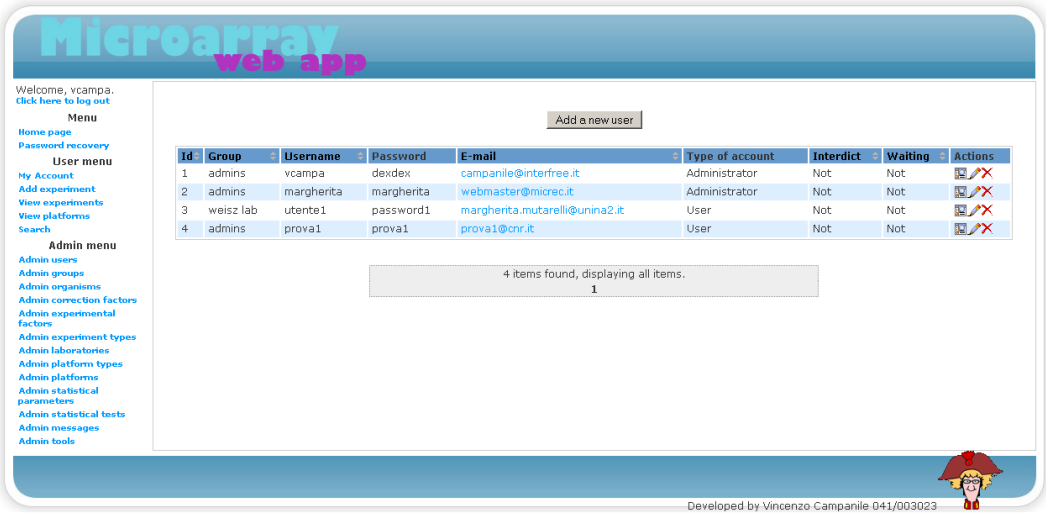

Figura 5.29.: Pagina di amministrazione degli utenti. Lo smartlinking di DisplayTag trasforma automaticamente in un link gli indirizzi di posta. I pulsanti a destra servono alle consuete operazioni di CRUD.

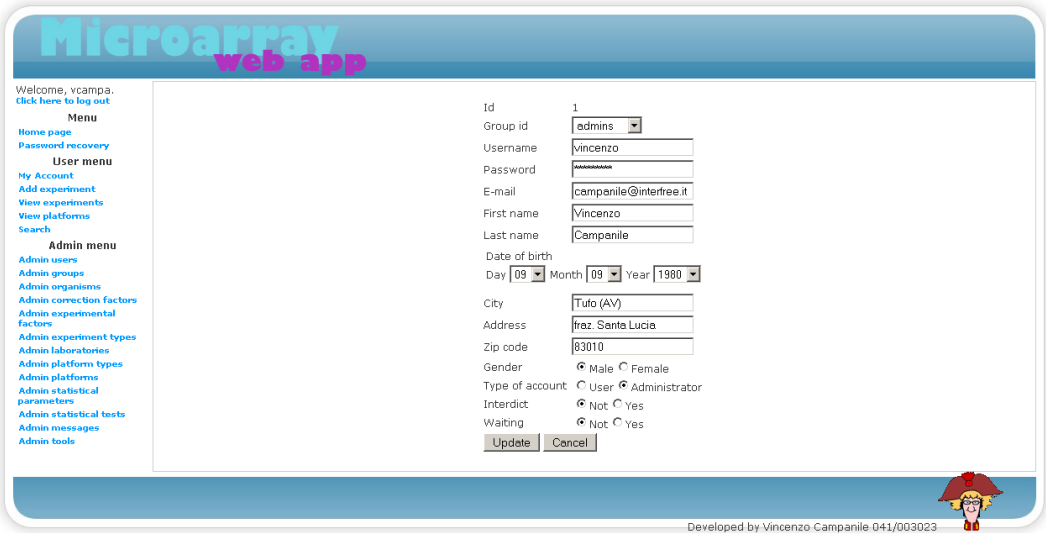

Figura 5.30.: Pagina per la modifica di un utente. La pagina include la possibilità di alterare lo stato dell'utente relativamente alla registrazione, di interdirlo, di modificarne il gruppo di appartenenza o di promuoverlo ad amministratore.

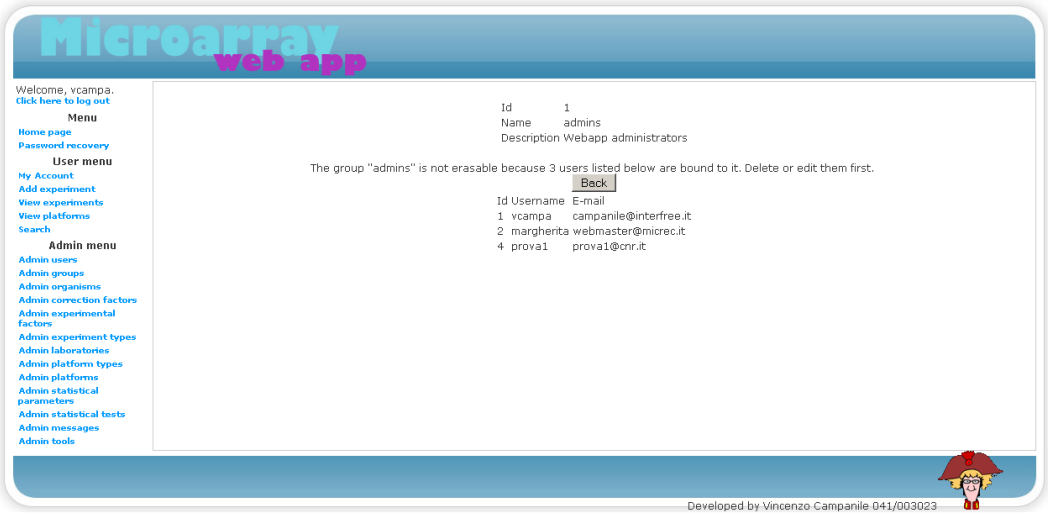

Figura 5.31.: Pagina per l'eliminazione di un gruppo. La pagina, oltre ad includere i dettagli relativi al gruppo di cui si è richiesta l'eliminazione, informa l'amministrazione che non è possibile eliminarlo se prima non si sono eliminati o modificati gli utenti ad esso afferenti.

6 Conclusioni

Alla fine di questo lavoro possiamo analizzare i risultati ottenuti, al fine di valutare fino a che punto siamo riusciti ad avvicinarci ai requisiti; quale è la qualità delle soluzioni adottate e quali sono gli sviluppi possibili attorno a questo progetto. Cercheremo di identificare i punti deboli di questo lavoro, la capacità di scalare e le problematiche che potrebbero nascere dall'utilizzo di questa applicazione al variare degli scenari.

Abbiamo sviluppato una applicazione che fa uso di una architettura solida, nonché ampiamente utilizzata, e delle tecnologie più recenti. Apache Struts è il veterano dei framework MVC ed è tutt'oggi il prodotto più riverito ed imitato. MySQL è il più celebre tra i RDBMS, universalmente noto per le performance offerte e per essere stato utilizzato con successo in contesti caratterizzati dalla necessità di gestire tabelle di milioni di record. Apache Lucene è tra i prodotti software dotati del maggior numero di port verso altri linguaggi.

Il driver JDBC consente di accedere a una base di dati residente sulla stessa macchina che ospita il servlet container, come accade sul server che abbiamo avuto a disposizione, ma consente anche l'accesso, senza dover rimaneggiare il codice, ad un database server dedicato. Ciò ci rende certi del fatto che, a patto di avere a disposizione le risorse economiche per dotarsi di infrastrutture più performanti, il sistema è in grado di scalare al crescere della mole di dati archiviati.

La soluzione adottata per la gestione dell'indice di Lucene, pur non consentendo la definizione in modo dichiarativo dell'origine dei dati, come accade per il driver JDBC, ci consente comunque di stabilire la locazione dell'indice sulla macchina che ospita l'applicazione. Al crescere dei dati è quindi possibile trasferirne i file su dischi dedicati, al limite, apportando qualche modifica al codice, è anche possibile spostare l'indice su una macchina dedicata o perfino distribuirlo su una rete.

Ad ogni modo, riteniamo che la ridondanza introdotta nei dati per via dell'indicizzazione limiti notevolmente le prospettive di scaling dell'applicazione. Nella versione proposta l'applicazione necessita di una duplicazione dei dati archiviati al fine di espletare le funzionalità di ricerca offerte. Pensiamo che sia possibile archiviare i dati nel solo indice, a patto di risolvere il problema della ricostruzione degli stessi, essendo questi memorizzati per righe, e di approfondire la tematica della sicurezza dell'indice come strumento di persistenza.

La compressione on the fly dei dati immessi, tanto nel database quanto nell'indice, ha consentito di annullare l'impatto che la ridondanza introdotta nei dati ha sui requisiti di sistema in termini di spazio su disco. Pur penalizzando le prestazioni a runtime del sistema, riteniamo che questa soluzione sarebbe stata utile anche in assenza di ridondanza. Infatti ci sembra che l'ampiezza dei dati da archiviare e la natura testuale degli stessi concorrano comunque nello sconsigliare la memorizzazione in chiaro.

Una ulteriore considerazione andrebbe fatta sullo scenario di utilizzo del sistema. L'applicazione è stata pubblicata sul web e quindi aperta al contributo di una platea di utenti dislocata in ogni punto del pianeta, ma è realmente funzionale? Partiti dal presuposto che la disponibilità di una connessione a banda larga è un requisito essenziale al trasferimento su rete di dati della dimensione di quelli gestiti dal sistema, ci chiediamo quanto tempo l'utente sia disposto ad attendere per l'immissione dei propri dati.

L'archiviazione, la compressione e l'eventuale indicizzazione richiedono una attesa di qualche minuto, che aggiunta a quella del trasferimento dei dati sulla rete, può risultare seccante. Per questo l'utilizzo dell'applicazione nell'ambito di una rete locale, dove il trasferimento dei dati impiega un tempo trascurabile, sarà sicuramente più confortevole rispetto all'utilizzo sul web. Se aggiungiamo le considerazioni sui requisiti hardware a queste ultime è facile convincersi che il sistema prodotto è più utile se confinato nelle strutture di un centro di ricerca.

Nella versione attuale, a parte l'applet VIMIDA per la visualizzazione dei dati normalizzati, il nostro sistema non offre funzionalità di analisi ed elaborazione dei dati. Simili funzionalità sono offerte dai tool forniti a corredo degli array, oltre che dai programmi di elaborazione normalmente utilizzati dai bioinformatici. Per questo riteniamo che l'integrazione, tramite applet, di un supporto alla rielaborazione dei dati lato client possa ampliare le prospettive di utilizzo del sistema.

L'applicazione accetta in input semplici file di testo. Ciò la rende estremamente generica e consente l'efficace applicazione degli algoritmi di indicizzazione di Lucene. Comunque riteniamo che possa essere più utile ai ricercatori una applicazione che accetti in ingresso i formati che essi utilizzano nel lavoro quotidiano, in particolare i fogli di calcolo. Ciò ne avvantaggerebbe l'accessibilità senza comprometterne la genericità. Sono disponibili, infatti, API in Java per la manipolazione, e quindi la conversione in testo semplice necessaria ai fini dell'indicizzazione, dei formati più diffusi.

L'indicizzazione dei dati è limitata, allo stato attuale alle sole tabelle delle annotazioni; come si è spiegato nel capitolo sull'indicizzazione, sono stati indicizzati solo gli identificatori di riga dei dati normalizzati e delle liste di geni regolati. Ciò non consente la ricerca su range di valori numerici che comunque Lucene supporta. Abbiamo addotto motivi prestazionali a questa scelta, ma riteniamo

che una estensione delle funzionalità dell'applicazione possa lavorare in questo senso.

Una serie di parametri di configurazione, come la locazione dell'indice, gli account da utilizzare per spedire messaggi, il numero di risultati visualizzati per pagina, l'ora a cui lanciare l'indicizzazione batch, sono stati memorizzati, con un abuso funzionale, nel resource bundle ed acceduti tramite MessageResources. Ciò ne rende possibile la modifica senza la necessità di ricompilare il codice dell'applicazione. Sarebbe stato sicuramente più opportuno utilizzare per la configurazione un bundle separato da quello utilizzato per l'internazionalizzazione dei messaggi, o anche un file di configurazione XML.

Per quanto riguarda la logica funzionale implementata possiamo dire con certezza che l'applicazione è autosufficiente. Funzionalità di CRUD complete sono state associate a tutti i dati gestiti, così è possibile leggere, aggiungere, modificare ed eliminare i dati archiviati nel database e le voci indicizzate, comodamente dall'interfaccia web, senza dover ricorrere a tool di terze parti. Il layout del portale consente una navigazione semplice, veloce ed intuitiva, in linea con l'accessibilità offerta dalle più comuni soluzioni web.

7 Riferimenti

Capitolo 1 - Introduzione

- Programming Jakarta Struts ,Chuck Cavaness - O'Reilly, 2003
- J2EE Design and Development, Rod Johnson - Wrox, 2002
- Minimum Information About a Microarray Experiment - MIAME 1.1 Draft 6
  http://www.mged.org/Workgroups/MIAME/miame_1.1.html
- Minimum information about a microarray experiment (MIAME) - toward standards for microarray data, - Nature Publishing Group, 2001
- Sviluppare applicazioni J2EE con Jakarta Struts, Alfredo Larotonda - MokaByte 81, 01/2004
  http://www.mokabyte.it

Capitolo 2 – Modelling dei dati

- Basi di dati, seconda edizione, P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone – McGraw-Hill, 2006
- Sybase Power Designer: User's Guide - Sybase
  www.sybase.it
- SQL For Dummies, 5th Edition, Allen G.Taylor - Wiley Publishing, 2003

- Database Design for Smarties: Using UML for Data Modeling, Robert J. Muller - Morgan Kaufmann Publishers, 1999
- MySQL 5.0 Reference Manual, MySQL AB 2007 www.mysql.com

Capitolo 3 – Progettazione dell'applicazione

- Iterative and incremental development, Wikipedia, the free encyclopedia http://en.wikipedia.org
- Principi d'ingegneria del software, Roger S. Pressman - Mc-Graw-Hill, 2000
- UML 2 for Dummies, Michael Jesse Chonoles and James A. Schardt - Wiley Publishing, 2003
- UML e Ingegneria del Software, dalla Teoria alla Pratica, Luca Vetti Tagliati – Hops, 2003
- Mastering Jakarta Struts, James Goodwill - Wiley Publishing, 2002
- Creating Use Case Diagrams, Mandar Chitnis, Pravin Tiwari, Lakshmi Ananthamurthy – Developer.com http://www.developer.com
- Mapping use cases to Struts – Lalilуna http://www.lalilуna.de

Capitolo 4 – Indicizzazione e ricerca

- Information retrieval - Wikipedia, the free encyclopedia http://en.wikipedia.org

- Modern Information Retrieval: A Brief Overview, Amit Singhal - Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2001
- Lucene in Action, Erik Hatcher, Otis Gospodnetic – Manning, 2005
- Professional Portal Development with Open Source Tools: Java™ Portlet API, Lucene, James, Slide, W. Clay Richardson, Donald Avondolio, Joe Vitale, Peter Len - Kevin T. Smith - Wiley Technology Publishing
- Search Enable Your Application With Lucene - Java Developer's Journal, 12/2002
- The Lucene search engine: Powerful, flexible, and free – JavaWorld
  http://www.javaworld.com
- Using Lucene to Search Java Source Code, Renuka Sindhgatta – ONJava.com
  http://www.onjava.com

Capitolo 5 – Implementazione dell'applicazione

- Jakarta Struts For Dummies, Mike Robinson and Ellen Finkelstein - Wiley Publishing, Inc, 2004
- Jakarta Struts Cookbook, Bill Siggelkow – O'Reilly, 2005
- Struts Tutorial by Stephan Wiesner - Stephan Wiesner, 2002
- Struts: The Complete Reference, James Holmes – McGraw-Hill/Osborne, 2004
- Programming Jakarta Struts ,Chuck Cavaness - O'Reilly, 2003
- Struts, JSP and Servlets Tutorials – Lalilуna
  http://www.lalilуna.de/struts-jsf-jsp-servlets-tutorials.html

- Struts Survival Guide: Basics to Best Practices, Srikanth Shenoy and Nithin Mallya – ObjectSource, 2004
- Struts Kick Start, Kevin Bedell and James Turner - Sams Publishing, 2003
- Struts in Action: Building web applications with the leading Java framework, Ted Husted – Manning, 2003
- Struts best practices, Puneet Agarwal – JavaWorld.com, 2004 http://www.javaworld.com
- Struts FAQ – JGuru http://www.jguru.com
- Java 2 Platform Standard Edition 5.0 API Specification - Sun Microsystems, Inc. http://java.sun.com/j2se/1.5.0/docs/api
- Java Platform Enterprise Edition, v 5.0 API Specifications - Sun Microsystems, Inc. http://java.sun.com/javaee/5/docs/api
- Struts User's Guide – Apache Software Foundation http://struts.apache.org/1.2.9/userGuide
- Quartz Documentation – OpenSymphony http://www.opensymphony.com/quartz
- Lucene Documentation – Apache Software Foundation http://lucene.apache.org/java
- Commons Email User's Guide – Apache Commons http://commons.apache.org/email
- DisplayTag library Documentation http://displaytag.sourceforge.net/11
- VIMIDA – A. Zinovyev http://bioinfo-out.curie.fr/projects/vimida