Università degli Studi di Napoli “Federico II”

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

TESI DI LAUREA

Progetto e realizzazione di una applicazione web-based sulle proteine

Relatore:

Ch.mo Prof. Antonio d’Acierno

Correlatore:

Ch.mo Dott. Angelo Facchiano

Candidato:

Michele Festa

Matr.: 041/2835

ANNO ACCADEMICO 2004/2005

A mio nonno Michele,
a mia madre Anna e
a mio padre Mario
…grazie di tutto!!!

INDICE

CAPITOLO UNO

INTRODUZIONE

Questo capitolo introduttivo illustra il problema affrontato e descrive gli obiettivi della tesi. Si procede ad una descrizione a grandi linee del contenuto dei capitoli successivi e dell'architettura del sistema. Il materiale presentato è dunque di fondamentale importanza per la comprensione di tutto il seguito del testo.

1.1 Premessa

Oggetto di questa tesi è la descrizione del processo di progettazione e realizzazione di una applicazione web per la memorizzazione, la gestione e la consultazione dei dati relativi alle proteine. In particolare, si vuole mettere in evidenza la struttura di varie proteine e le mutazioni che possono modificare tale struttura, comportando le cosiddette “malattie metaboliche ereditarie”.

La tesi tratta principalmente gli aspetti informatici del progetto, mentre vengono fornite solo le informazioni chimiche e biochimiche strettamente necessarie alla comprensione dello stesso.

1.2 Presentazione del Problema

E’ sempre più frequente sentir parlare di persone, e in particolare bambini nei primi mesi di vita, che soffrono di particolare intolleranze a determinati alimenti, come ad esempio il latte. Molte di queste intolleranze sono dovute a *Malattie Metaboliche Ereditarie* [1] [2] [6].

Con il termine *metabolismo* si indica l’insieme dei processi che l’organismo mette in atto per trasformare le proteine, i grassi e gli zuccheri, contenuti negli alimenti, in altri composti utili oppure in sostanze più semplici, allo scopo di ricavare energia, che viene utilizzata dalle cellule per svolgere le proprie funzioni biologiche. Questi processi sono generati da reazioni chimiche, che vengono regolate da alcuni enzimi o

proteine con funzioni enzimatiche. Le malattie metaboliche si manifestano quando manca o è malfunzionante uno di questi enzimi, comportando da una parte un accumulo di sostanze, spesso tossiche per l'organismo, e dall'altra una riduzione di energia per le cellule. Tali deficienze si ripercuotono sulla struttura della proteina provocandone una mutazione.

Queste disfunzioni sono di natura ereditaria, in particolare le statistiche indicano che da due genitori, portatori sani della malattia, nasce un figlio affetto da malattia metabolica con una probabilità del 25%.

Il tipo di sintomi e gli organi colpiti variano in funzione del tipo di danno: ogni malattia metabolica ereditaria ha quindi caratteristiche cliniche diverse. L'unica cosa certa è che l'entità dei danni provocati è proporzionale al tempo impiegato per diagnosticare la malattia. Una diagnosi ritardata può provocare handicap psico-motori ed in alcuni casi può essere anche letale.

Al giorno d'oggi non esiste una cura che permette una guarigione definitiva, anche se si prevede che in un futuro non molto lontano possano essere individuate terapie geniche a tal proposito. Le terapie che vengono attualmente utilizzate prevedono l'eliminazione dalla dieta dell'alimento non tollerato dall'organismo e l'assunzione di farmaci in grado di facilitare la depurazione dell'organismo dai prodotti tossici [3] [5].

L'applicazione che ci accingiamo a presentare non si propone di illustrare le malattie metaboliche, in quanto ciò rappresenterebbe solo una ripetizione di informazioni già ampiamente disponibili sul Web. Essa si pone come obiettivo primario

quello di mettere a disposizione informazioni utili alle persone coinvolte in studi biochimici sugli enzimi che comportano le malattie suddette. Saranno rese disponibili le informazioni sulle mutazioni che possono verificarsi nelle proteine, ma anche i risultati di una serie di analisi utili ad analizzare gli effetti che tali mutazioni comportano sulla struttura di tali proteine.

1.3 Definizioni Fondamentali

Prima di descrivere a fondo la natura del progetto, si ritiene necessario dare alcune definizioni basilari riguardanti l'ambito applicativo di riferimento. Per maggiori informazioni si faccia riferimento alla letteratura specifica e alla bibliografia presentata [4] [9].

Si tenga comunque presente che le definizioni presentate non vogliono essere delle definizioni tecniche rigorose, ma solo un semplice aiuto per coloro che si accingeranno a proseguire la lettura.

Proteina: Molecola complessa composta di aminoacidi necessaria per i processi chimici che si presentano negli organismi viventi;

Aminoacido: piccola molecola che si lega con altre molecole simili in lunghe catene per formare le proteine;

Molecola: è una collezione di atomi legati chimicamente, con una particolare composizione e struttura;

Catena della proteina: Struttura primaria della proteina, è la sequenza con la quale gli aminoacidi si susseguono nella proteina. Sostituendo anche solo uno degli aminoacidi che la costituiscono con un altro, si modificano i caratteri chimici della proteina. Ciascun elemento(aminoacido) della catena viene anche indicato con il termine residuo;

Numero di residuo: numero rappresentante l'ordine degli aminoacidi all'interno della proteina;

Nucleotide: uno dei componenti strutturali del DNA e del RNA(Es. Timina, Guanina, ecc.);

Codon: tripletta di nucleotidi che codificano un aminoacido;

Mutazione Missense: mutazione di aminoacido dovuta al cambio di uno o più nucleotidi nel codon, che comporta la codifica di un aminoacido diverso da quello originario;

Enzima: sostanza organica capace di modificare la velocità di alcune reazioni chimiche e biochimiche;

Indice di conservazione del residuo: rappresenta il grado di conservazione dell'aminoacido, all'interno della catena della proteina, in altre specie differenti dall'uomo;

Analisi con DSSP, **Analisi con NACCESS su proteina dimerica**,**Analisi con NACCESS su proteina monomerica**, **Analisi con HBPLUS sui legami idrogeno**, **Analisi interazioni proteina-ligando**, **Legami idrogeno proteina-ligando**: analisi effettuate per avere ulteriori informazioni sulla struttura della proteina, sulla posizione degli aminoacidi, sui legami che gli aminoacidi formano con l'idrogeno o con altri elementi.

1.4 Linee Guida del Progetto

Nelle fasi iniziali del progetto fissiamo dei punti chiave, l'applicazione dovrà:

1. essere accessibile da due tipi di utenti, un amministratore e vari utenti generici;

2. essere accessibile via web tramite un'interfaccia amichevole, sia nella parte amministrativa privata che nella parte pubblica;

3. essere in grado di identificare l'amministratore e autorizzarlo ad accedere alla parte amministrativa privata, vietando l'accesso a chiunque altro;

4. garantire la sicurezza delle informazioni trattate nella parte amministrativa privata;

5. contenere un supporto multilingua, ovvero le informazioni devono essere consultabili in lingua inglese ed in lingua italiana, predisponendo l'eventuale aggiunta di ulteriori lingue in futuro;

6. essere facilmente gestibile e manutenibile.

Come si può vedere, queste semplici linee guida sono ben lungi dall'essere un rigoroso documento di specifiche: questo perché scopo della tesi è descrivere il progetto e non documentarlo approfonditamente, come invece si cerca di fare con il materiale allegato alla tesi stessa.

1.5 Passi Fondamentali

Nello svolgere il progetto sono stati seguiti gli step riportati di seguito (ovviamente la presentazione in serie dei passi non è indicativa di un processo di sviluppo sequenziale):

1. Valutazione delle tecnologie utilizzate per la progettazione e realizzazione dell'applicazione web: questo step comprende l'analisi dei vari metodi di sviluppo di un prodotto software per il Web e lo studio delle varie tecnologie per la progettazione e realizzazione di applicazioni web, con l'obiettivo di selezionare la tecnologia che sia più idonea alle nostre necessità;

2. Ricerca e selezione delle informazioni utili al progetto;

3. Progettazione della base di dati: si analizzano le informazioni selezionate al fine di individuare quelle necessarie per la costruzione di un modello concettuale adatto all'applicazione da realizzare, per poi effettuare la trasformazione nel modello relazionale e quindi il modello fisico che verrà utilizzato;

4. Progettazione della applicazione web, in base alla metodologia progettuale selezionata;

5. Realizzazione della applicazione web, in base alla tecnologia scelta;

6. Testing dell'applicazione web, al fine di individuare il maggior numero possibile di malfunzionamenti e/o errori.

1.6 Architettura del Sistema

L'architettura del sistema è una architettura classica Client/Server (Fig. 1.1), caratterizzata dalla presenza di un server centrale (localizzato nell'Istituto di Scienze dell'Alimentazione(ISA) presso il Centro Nazionale di Ricerche(CNR) di Avellino (*www.isa.cnr.it*) e dalla presenza di due tipi di Client, Client per la gestione dei dati (amministratore) e Client per la consultazione dei dati (utenti). Ovviamente la suddivisione dei Client in due categorie è solo esemplificativa, in quanto, ad esempio,

l'amministratore può essere anche un utente e quindi effettuare la consultazione dei dati.

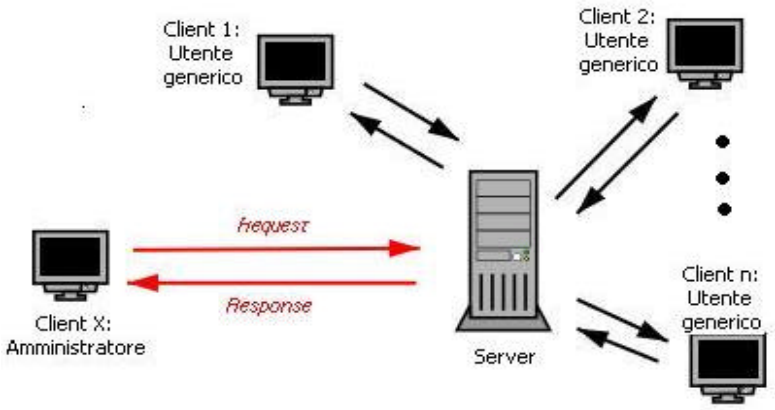


Figura 1.1 - Architettura Client/Server

CAPITOLO DUE

SELEZIONE E DESCRIZIONE DELLA METODOLOGIA E DELLE TECNOLOGIE DI SVILUPPO

Questo capitolo, dopo una breve introduzione sullo sviluppo di applicazioni Web, illustra il ciclo di sviluppo software e le tecnologie di ausilio alla progettazione utilizzati. In seguito viene effettuata una breve analisi delle metodologie realizzative conosciute, selezionando quella ritenuta migliore.

2.1 Introduzione allo Sviluppo di Applicazioni Web

Prima di entrare nel dettaglio delle scelte relative alla metodologia progettuale e alle tecnologie utilizzate nelle fasi di progettazione e di implementazione, è preferibile illustrare in breve gli aspetti salienti dello sviluppo di una applicazione Web.

Innanzi tutto è da sottolineare la differenza tra:

- *Sito Web*, software che mostra ad un utente informazioni in una modalità di "sola lettura", senza cioè che esso possa interagire per modificare lo stato del software stesso ed in particolare i dati;

- *Applicazione Web*, software, utilizzante la stessa infrastruttura di un sito, che consente ad un utente di interagire con esso permettendo la modifica dello stato del software.

Il sistema che verrà progettato e realizzato nell'ambito di questa tesi è, come più volte detto, una applicazione web. Questo perché sia l'utente amministratore che l'utente generico devono interagire con i dati presenti nella banca dati.

Nel corso degli anni Internet e il World Wide Web[1] hanno avuto una crescita e un'accettazione oltre ogni più rosea aspettativa. Questa crescita ha generato la necessità di introdurre nuovi modelli e metodi di sviluppo, specifici per il Web.

[1] World Wide Web (WWW), sistema su scala planetaria per la distribuzione e l'accesso a documenti ipertestuali e multimediali codificati sotto forma di pagine HTML.

Numerose metodologie sono state pensate appositamente per i siti web e, in seguito, per le applicazioni web, come ad esempio:

- RMM (Relationship Management Methodology);
- OOHDM (Object-Oriented Hypermedia Design Method);
- WEBML (Web Modeling Language).

Ovviamente, come è accaduto per i prodotti software classici, il grande e rapido sviluppo delle applicazioni Web ha fatto sorgere la necessità di avere un linguaggio unificato per la progettazione. Ciò ha portato all'adattamento del linguaggio UML [19] [21] per la modellazione di applicazioni Web. Il documento di riferimento per tale adattamento è il libro di J. Conallen [20].

Nella Figura 2.1 sono rappresentate le estensioni di UML

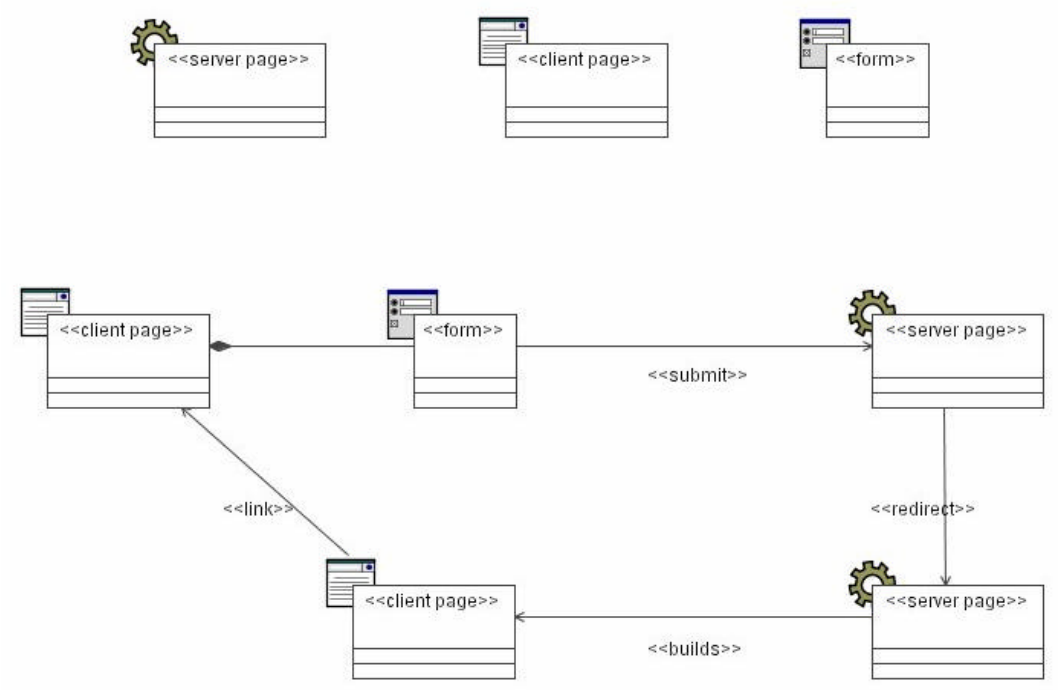


Figura 2.1 - Estensioni del linguaggio UML per le applicazioni Web

Come si può notare sono state aggiunte:

- le classi: Client Page, Server Page e Form;
- le associazioni: Link, Submit, Builds, Redirect.

2.2 Descrizione della Metodologia di Sviluppo

La metodologia di sviluppo [22] utilizzata per la realizzazione del software risultante da questa tesi è suddivisibile nei seguenti passi:

1. Analisi e specifica dei requisiti;

2. Progettazione dell'applicazione

2.1 Individuazione degli Use Case, utili per la visualizzazione degli attori che opereranno con l'applicazione e dei casi d'uso che rappresentano le modalità di utilizzo del sistema da parte degli attori, senza specificare la "logica interna" delle varie operazioni effettuate;

2.2 Costruzione del Modello Concettuale dei dati, con conseguente trasformazione in modello logico e quindi fisico;

2.3 Costruzione dei Class Diagrams, in cui ogni classe rappresenta una pagina visualizzata e le associazioni rappresentano le interazioni tra le pagine;

2.4 Studio dinamico e costruzione dei Sequence Diagrams, utilissimi per le applicazioni Web in quanto specificano come viene costruita la pagina per l'utente e inoltre per ogni caso d'uso indicano anche i passi che l'utente deve seguire per giungere al risultato desiderato;

3. Implementazione dell'applicazione;

4. Testing dell'applicazione;

Al termine di tale ciclo di sviluppo avviene il rilascio dell'applicazione. Ovviamente la sequenzialità tra le varie fasi non è rigida, bensì è possibile che in una delle fasi ci si renda conto di errori nelle fasi precedenti e quindi si ritorna indietro, per

eliminarli. Anche dopo il rilascio dell'applicazione è possibile che gli utenti dell'applicazione segnalino alcuni malfunzionamenti, che potrebbero essere relativi a fasi precedenti e che dovranno essere eliminati, pertanto è prevista anche una fase di manutenzione post-rilascio.

Possono risultare utili anche altri tipi di diagrammi UML, come ad esempio gli Activity Diagram nel caso in cui nell'applicazione che si vuole realizzare sono presenti funzioni complesse, non rappresentabili chiaramente con i diagrammi precedenti.

Scopo della tesi non è la descrizione dettagliata della fase di progettazione dell'applicazione, per cui verrà effettuata solo una presentazione del processo. Per maggiori dettagli consultare l'Appendice B in cui sono riportati tutti i diagrammi e le informazioni relative ai passi 2, 4 e 5 descritti in precedenza. Per ulteriori informazioni viene fornito del materiale allegato in cui sono presenti i documenti di tutte le fasi sviluppate.

2.3 Tecnologie di Supporto alla Progettazione

Per la progettazione dell'applicazione Web è stato utilizzato il prodotto CASE (Computer Aided Software Engineering) **PowerDesigner,** uno strumento flessibile e con supporto per la stragrande maggioranza dei linguaggi di programmazione e dei DBMS presenti sul mercato. Tale prodotto, ottimo per la realizzazione del diagramma dei Casi d'Uso e per la progettazione del modello concettuale e fisico della base di dati, si è però rivelato inadatto per la realizzazione dei diagrammi UML dell'applicazione in

quanto è risultato essere privo delle estensioni UML per il Web, di cui si è brevemente parlato in precedenza. Per tale motivo è sorta la necessità di utilizzare un altro pacchetto software per la realizzazione di tali diagrammi. In particolare tra le varie opportunità disponibili si è deciso di utilizzare il prodotto **MagicDraw UML**, in possesso di tutti i requisiti per la progettazione dell'applicazione Web.

2.4 Tecnologie Utilizzate per l'Implementazione

Il sistema operativo presente sul server, presso il CNR, dove dovrà essere installata l'applicazione è **Linux** (in particolare la distribuzione Red Hat 9). Ciò soprattutto per motivi di sicurezza e di affidabilità: le alternative (i sistemi Microsoft) sono purtroppo rinomate per essere più vulnerabili. Per il nostro obiettivo, tale "scelta obbligata" si è rivelata molto conveniente, in quanto trattandosi di un progetto svolto nell'ambito di una tesi, si è sempre cercato di utilizzare software freeware (o al massimo shareware): in ambiente Linux è molto più facile reperire questo tipo di applicativi.

I server da realizzare sono ovviamente un DBMS server ed un web server (ovviamente risiederanno entrambi sulla stessa macchina). La tecnologia utilizzata per il primo è **PostgreSQL** [16]: si tratta di uno tra i migliori DBMS freeware disponibili su piattaforma Linux/Unix, con caratteristiche simili a ben più blasonati prodotti commerciali.

Per l'accesso ai dati è stata utilizzata la tecnologia JDBC (Java DataBase Connectivity), ed in particolare la libreria specifica del DBMS utilizzato, facilmente reperibile presso il sito http://jdbc.postgresql.org.

E' inoltre stato realizzato un prototipo per ambiente Windows utilizzando Microsoft Access come DBMS, per valutare la validità e le prestazioni del framework utilizzato per la realizzazione dell'applicazione Web in un altro ambiente. Per tale prototipo si è utilizzata la tecnologia di accesso ai dati basata su ODBC (Open DataBase Connectivity).

Per quanto riguarda il Web Server, la scelta è caduta su **Apache Tomcat** [14], realizzato nell'ambito del progetto Jakarta[2][13] dalla fondazione Apache[3][12], fra i più sicuri e testati per questo genere di applicativi.

In realtà Tomcat nasce come Application Server[4], molto spesso utilizzato in combinazione con il Web Server[5] Apache. E' comunque possibile utilizzare Tomcat come Web Server; nel progetto in esame si è preferita questa soluzione perchè Tomcat funge anche da Container per applicazioni web, conforme alle specifiche della piattaforma Java 2 Enterprise Edition(J2EE) (v. Appendice C). Un *Container*, in generale, fornisce un ambiente che ospita componenti software che vanno in esecuzione all'interno di tale "contenitore". Esso rende disponibili dei servizi generali che i componenti all'interno di tale ambiente possono utilizzare. In particolare, Tomcat permette di effettuare il deploy e di eseguire al suo interno servlet, JSP e altre classi Java.

---

[2] *Apache Jakarta* è un progetto nato in seno alla Apache Software Foundation che offre un'insieme di soluzioni e librerie basate su Java. E' in realtà un contenitore di altri sotto-progetti quali Tomcat, Struts, ecc.

[3] L'*Apache Software Foundation* è una fondazione non-profit ed una comunità distribuita di sviluppatori che lavorano su progetti open-source. Ciascun progetto è gestito da un team di volontari che sono i contributori attivi al progetto.

[4] Un *Application Server* è un'applicazione che fornisce l'infrastruttura e le funzionalità di supporto a integrazione, deploy ed esecuzione di applicazioni e componenti server in un contesto distribuito.

[5] Un *Web Server* è un programma che riceve le richieste http dal browser e restituisce la pagina da visualizzare tramite una risposta http.

Per la scelta della tecnologia realizzativa sono state effettuate alcune considerazioni, al fine di individuare la metodologia più idonea alla realizzazione dell'applicazione.

Nel corso degli anni sono state sviluppate e standardizzate varie tecnologie per la realizzazione di pagine dinamiche, ciascuna con i propri pregi e difetti. Analizziamo le varie possibilità:

1. *Common Gateway Interface* (*CGI*), è il primo e più semplice standard architetturale per la creazione di pagine dinamiche. Il principio di funzionamento è molto semplice, il Web Server riceve una richiesta http da un browser(che identifica uno script CGI), salva le informazioni interne alla richiesta in memoria, riconosce qual è lo script CGI richiesto e lo invoca. Lo script CGI può contenere interrogazioni a qualche base di dati oppure solo elaborazioni che portano alla produzione dinamica della pagina HTML che verrà inviato al Web Server, il quale a sua volta utilizzerà la pagina per costruire la risposta http da inviare al browser. A fronte del vantaggio di poter costruire pagine dinamicamente, ci sono alcune serie limitazioni, come ad esempio l'enorme esigenza di risorse. Infatti per ogni richiesta http viene creato un nuovo processo del Sistema Operativo che viene terminato dopo la costruzione della pagina dinamica. Questo continuo avvio e distruzione di processi produce un elevato sovraccarico delle prestazioni del sistema. Altro notevole limite è dovuto al fatto che la terminazione del processo CGI dopo l'evasione della richiesta non consente di avere a disposizione in memoria centrale strutture dati che possano essere condivise tra diverse richieste. Tali limitazioni hanno portato all'ideazione di architetture più potenti;

2. *Servlet Java*, forniscono un metodo component-based e indipendente dalla piattaforma, per costruire web application. Esse creano un solo processo e consentono a ciascuna richiesta utente di utilizzare un thread più leggero, che viene gestito dalla Java Virtual Machine(JVM[6]). Ciascuna richiesta è associata con un thread separato, quindi thread o utenti multipli possono invocare lo stesso metodo contemporaneamente. Un vantaggio significativo è dato dal fatto che essendo scritte in Java possono sfruttare l'intera e ricca serie delle Application Programming Interfaces (API) Java, tra cui JDBC ed EJB, e delle libreria Java. Le servlet non vengono direttamente eseguite da un Web server, ma hanno bisogno di un contenitore, servlet container, che le ospiti. Il web server e il servlet container lavorano insieme per soddisfare le richieste. Esistono molti container disponibili, quindi le servlet non dipendono da un particolare produttore o da una specifica piattaforma. A fronte di tutti questi vantaggi risultano anche degli svantaggi, dovuti principalmente al fatto che le servlet devono incorporare stabilmente output HTML. Se, dopo aver sviluppato un sito di grandi dimensioni, si volesse cambiare l'estetica bisognerebbe rivedere l'intero codice e ricompilare la servlet. Un altro problema da non sottovalutare è legato alle responsabilità, perché i web designer costruiscono pagine HTML, ma non sempre sono a conoscenza di linguaggio Java, per cui questo mescolamento di Html e Java nelle servlet comporta dei tempi di sviluppo e verifica maggiori di altre soluzioni;

3. *Java Server Pages*(*JSP*), rappresentano la naturale estensione delle Servlet Java, infatti dopo una preventiva elaborazione vengono trasformate in servlet. Esse sono composte da codice HTML, con all'interno scriptlet JSP e tag. Viene

[6] Java Virtual Machine (JVM), è la macchina virtuale che esegue i programmi in linguaggio bytecode, ovvero i prodotti della compilazione dei sorgenti Java.

così risolto il problema dell'inglobamento del codice HTML nelle servlet, infatti può essere prima creata la grafica in HTML dai web designer e poi aggiunti gli scriptlet e i tag dagli sviluppatori JSP.

Per l'utilizzo della tecnologia JSP sono state ideate due architetture:

I. *Architettura JSP Model 1*, prevede che la pagina JSP gestisca tutta l'elaborazione della richiesta e ha la responsabilità di mostrare il risultato al Client, ovvero contengono al loro interno logica applicativa e logica di controllo del flusso;

II. *Architettura JSP Model 2*, prevede che la richiesta del Client venga prima intercettata da una servlet di controllo(o controller servlet) che gestisce l'elaborazione iniziale della richiesta e determina quale pagina JSP deve essere visualizzata come successiva. L'obiettivo è quello di avere nelle pagine JSP solo codice HTML e custom tag, con la minor presenza possibile di scriptlet, in quanto essi mischiano operazioni logiche e presentazione. La servlet di controllo è molto utile in quanto permette di effettuare controlli, quali ad esempio l'autenticazione e l'autorizzazione, prima di tutte le altre operazioni;

La differenza principale tra le due architetture sta nel fatto che nella architettura Model 2 c'è una chiara separazione tra business logic, presentation logic e control logic. Tale separazione viene spesso indicata come pattern *Model-View-Controller*, di cui si parla nel paragrafo seguente,

2.4.1 Il pattern Model-View-Controller

Il pattern MVC è una implementazione dell'architettura JSP Model 2, in esso è quindi presente una servlet di controllo che gestisce tutte le richieste e le soddisfa delegando l'elaborazione a opportune classi Java. In questo modo i ruoli di presentation, control e business logic vengono affidati a componenti diversi e tra di loro disaccoppiati, con notevoli vantaggi in termini di riusabilità, manutenibilità, estensibilità e modularità.

In un'applicazione costruita secondo il pattern MVC si possono quindi individuare tre livelli logici(Fig. 2.2) ben distinti che svolgono i seguenti compiti:

- *Controller*, si occupa di intercettare le richieste http, esaminarle, estrarne i parametri, tradurle in specifiche operazioni della business logic da effettuare ed infine di scegliere la successiva vista da fornire all'utente al termine dell'elaborazione;

- *Model*, può essere rappresentato da un database o da un JavaBeans, rappresenta ciò che viene elaborato e successivamente presentato all'utente;

- *View*, è la parte dell'applicazione che si occupa di visualizzare le informazioni all'utente. In un'applicazione web è la visualizzazione della pagina di risposta alla richiesta dell'utente.

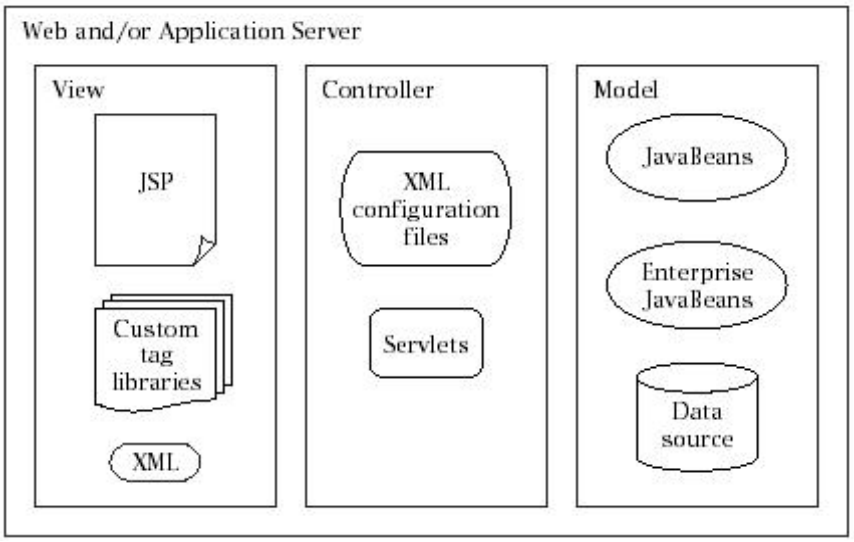


Figura 2.2 - Oggetti presenti nei livelli logici del pattern MVC

2.4.2 Selezione della Metodologia Realizzativa

Dopo aver valutato attentamente le varie possibilità, con i relativi vantaggi e svantaggi, per la realizzazione della applicazione oggetto della tesi si è deciso di adottare un framework che aderisca al design pattern MVC: il ***framework Struts***.

In particolare, l'ambiente di sviluppo che si è utilizzato per la realizzazione dell'applicazione Web è la piattaforma **Eclipse** [31] nella versione 3.0.2**,** un IDE(Integrated Development Environment) Java, un ambiente di sviluppo molto diffuso e distribuito con licenza open-source. La sua caratteristica principale è quella di avere una architettura a plug-in, il che lo rende un ambiente molto versatile ed adatto alle varie esigenze. Il plug-in scelto per lo sviluppo dell'applicazione è **Exadel Studio** [30] nella versione 2.5, il quale, oltre a supportare lo sviluppo di progetti con tecnologia Struts, fornisce una versione

dell'Application Server Tomcat, molto utile per testare velocemente l'applicazione, la possibilità di gestire graficamente la configurazione del progetto e la possibilità di utilizzare la scrittura assistita, per pagine JSP e XML.

Il grande sviluppo di Struts ha portato alla creazione di tantissimi plug-in per Eclipse con lo stesso obiettivo di Exadel, ma la nostra scelta è stata dettata principalmente dalla disponibilità di una versione freeware.

CAPITOLO TRE

IL FRAMEWORK STRUTS

Questo capitolo rappresenta una presentazione del framework Struts, ovvero il framework utilizzato per la realizzazione dell'applicazione Web, cercando di illustrare le caratteristiche che hanno portato al suo grande utilizzo per la realizzazione di applicazioni Web..

3.1 Introduzione al Concetto di Framework

Prima di iniziare la descrizione del framework specifico, si ritiene opportuno fornire delle definizioni generali, al fine di evitare confusione tra varie nozioni.

Un *framework* è una architettura generica che costituisce l'infrastruttura per lo sviluppo di applicazioni in una determinata area tecnologica, nel caso di Struts si tratta del mondo delle applicazioni web. Detto in maniera molto semplice, è un insieme di classi ed interfacce di base, che costituiscono l'infrastruttura di una applicazione.

In base a questa definizione è facile pensare erroneamente che utilizzare un framework equivalga ad usare una libreria di classi, mentre in realtà vi è una sostanziale differenza tra le due cose.

Una libreria di classi, quali ad esempio le classi di base del linguaggio Java, viene utilizzata dallo sviluppatore per svolgere determinate funzionalità; in questo caso il codice che noi scriviamo invoca il codice esistente per svolgere una certa funzione, ma il controllo del flusso applicativo rimane a nostro carico.

Adottare un framework significa invece attenersi ad un specifica architettura ovvero nella pratica estendere le classi del framework e/o implementarne le interfacce. In tal caso sono i componenti del framework che hanno la responsabilità di controllare il flusso elaborativo.

E' inoltre importante precisare la differenza tra un framework e un design-pattern. Un *design-pattern*, come il MVC, è una strategia di soluzione di un problema comune, è qualcosa di concettuale che prescinde dall'implementazione tecnologica. Un framework è invece qualcosa di concreto, è un insieme di componenti che può

essere usato per realizzare una applicazione; tali componenti possono essere sviluppati secondo un design-pattern.

Il framework Struts implementa molti dei design-pattern J2EE di uso comune, ciò a garanzia di una architettura valida e ben sperimentata.

3.2 Il Framework Struts

Struts è un framework open-source per lo sviluppo di applicazioni web, facente parte del progetto Jakarta della fondazione Apache [29], per la realizzazione di applicazioni Web sulla piattaforma J2EE(v. Appendice C) della SUN sviluppato per incoraggiare l'utilizzo di un tipo di architettura che si basa sul pattern MVC(model-view-control). In realtà tale framework solo inizialmente è stato inquadrato come un sotto-progetto di Jakarta, ma in breve, dato il suo sviluppo, è diventato un progetto a sé stante.

Esso viene definito *"MVC web application framework"*, in quanto è composto da un insieme di classi ed interfacce che costituiscono l'infrastruttura per costruire web application pienamente aderenti alla piattaforma J2EE, conformi al design pattern MVC.

I componenti fondamentali di struts sono:

- *ActionServlet*: e' la servlet di controllo centralizzata che gestisce tutte le richieste dell'applicazione;

- *struts-config.xml*: E' il file XML di configurazione di tutta l'applicazione. In questo file vengono definiti gli elementi dell'applicazione e le loro associazioni;

- *Action*: le Action sono le classi alle quali la ActionServlet delega l'elaborazione della richiesta;

- *ActionMapping*: contiene gli oggetti associati ad una Action nello struts-config.xml come ad esempio gli ActionForward, le proprietà, ecc.;

- *ActionForm*: gli ActionForm sono classi contenitori di dati. Vengono popolati automaticamente dal framework con i dati contenuti nelle request http;

- *ActionForward*: contengono i path ai quali la servlet di Struts inoltra il flusso elaborativo in base alla logica dell'applicazione;

- *Custom-tags*: Struts fornisce una serie di librerie di tag per assolvere a molti dei più comuni compiti delle pagine JSP, per cercare di evitare o quanto meno di ridurre il più possibile l'utilizzo di scriptlet.

La ActionServlet è la servlet di controllo di Struts. Essa gestisce tutte le richieste client e smista il flusso applicativo in base alla logica configurata. Si potrebbe definire come la "spina dorsale" di una applicazione costruita su Struts.

Tutta la configurazione dell'applicazione è contenuta nel file *struts-config.xml*. Questo file XML viene letto in fase di start-up dell'applicazione dalla ActionServlet e definisce le associazioni tra i vari elementi di Struts. Nello struts-config.xml, come vedremo più dettagliatamente in seguito, sono ad esempio definite le associazioni tra i

path delle richieste http e le classi Action associate alla richieste stesse, le associazioni tra le Action e gli ActionForm, che vengono automaticamente popolati dal framework con i dati della richiesta ad essi associata e passati in input alla Action. Contiene inoltre l'associazione tra la Action e le ActionForward , ovvero i path configurati nello struts-config.xml ai quali la ActionServlet redirigerà il flusso applicativo al termine della elaborazione della Action.

A tutto ciò va, ovviamente, aggiunto il file *web-xml*, che non è specifico di Struts, ma è utilizzato in tutte le applicazioni Web che fanno uso delle Servlet. In esso viene specificato al server qual è la servlet controller dell'applicazione.

3.3 L'implementazione del pattern Model-View-Controller in Struts

Nella Figura 3.1 è rappresentato schematicamente il flusso elaborativo nella logica di Struts:

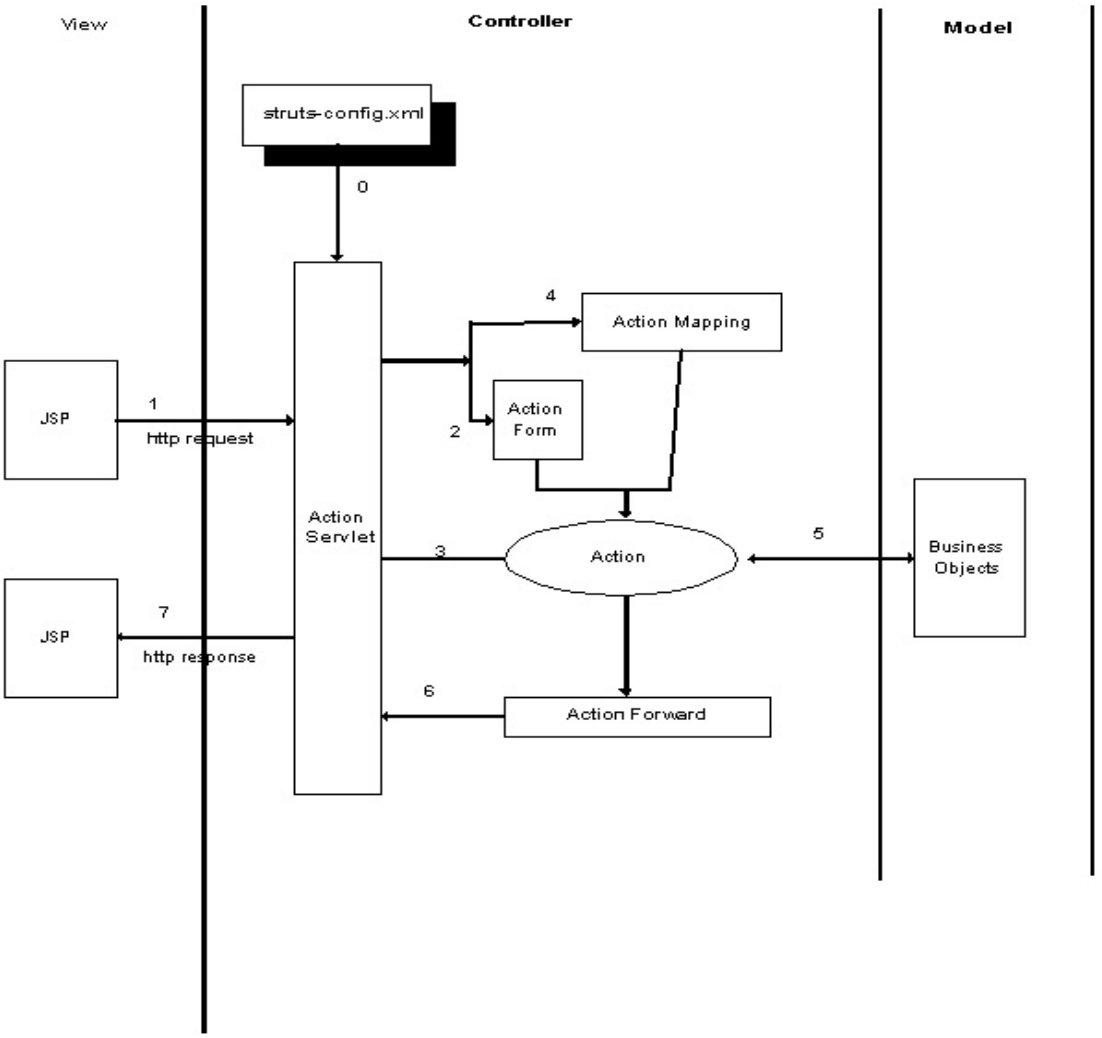


Figura 3.1 - Implementazione di Struts del pattern MVC

Analizziamo più nel dettaglio la sequenza di operazioni scatenate da una richiesta HTTP:

1. Il client invia una richiesta http tramite il proprio browser (1);

2. La richiesta viene ricevuta dalla servlet di controllo di Struts che provvede a popolare l'ActionForm, se presente, associato alla richiesta con i dati della request (2) e l'ActionMapping con gli oggetti relativi alla Action associata alla richiesta (4). Tutti i dati di configurazione sono stati letti in fase di start-up dell'applicazione (0) dal file XML struts-config.xml;

3. La ActionServlet delega l'elaborazione della richiesta alla Action associata al path della richiesta (3) passandole in input la request http, l'ActionForm e l'ActionMapping precedentemente valorizzati;

4. La Action si interfaccia con lo strato di business che implementa la logica applicativa. Al termine dell'elaborazione restituisce alla ActionServlet un ActionForward (6) contenente l'informazione del path della vista da fornire all'utente;

5. La ActionServlet esegue il forward alla vista specificata nell'ActionForward (7).

Ovviamente il flusso di operazioni elencato non è completo ma fornisce una indicazione di base su come viene gestito il flusso elaborativo in una applicazione sviluppata con Struts.

In realtà Struts non implementa interamente il pattern Model-View-Controller, esso è un *framework model-neutral*, ovvero implementa esclusivamente i livelli di controller e view. Utilizzando Struts è possibile realizzare il livello di business logic in base alle proprie scelte.

Descriviamo più dettagliatamente i due livelli implementati da Struts, mettendo in evidenza tutti i componenti e le operazioni da questi effettuate.

- **Componenti del Controller**

  Per un'applicazione costruita secondo l'architettura Model 2 delle JSP il controller è implementato per mezzo di una servlet Java. In Struts, la servlet di controllo è la classe *org.apache.struts.Action.ActionServlet*, la quale estende la classe javax.servlet.http.HttpServlet. Tale Servlet riceve sia richieste di tipo Get che di tipo Post ed effettua i seguenti step:

  1. in base alla richiesta invoca i metodi *doGet()* e *doPost()*, i quali, a loro volta, invocano il metodo *process()* della ActionServlet;

  2. nel metodo *process()* si ottiene l'istanza della classe; *org.apache.struts.Action.RequestProcessor*, di cui si esegue il metodo *process()*;

  3. nel metodo *process()* del Request Processor viene eseguita l'elaborazione vera e propria.

Il RequestProcessor viene configurato all'interno del tag <controller> del file struts-config.xml ed è possibile usarne uno proprio estendendolo e facendo l'override del metodo *processPreprocess()*; nel caso in cui non venga definito, viene naturalmente usato quello di default.

Il RequestProcessor esegue i seguenti step:

1. Legge il file struts-config.xml per trovare un elemento <action> corrispondente al path della richiesta;

2. Una volta trovato l'elemento <action> corrispondente verifica se è presente l'attributo name che corrisponde al nome dell'ActionForm configurato per la richiesta in elaborazione. In tal caso provvede a reperire una istanza dell'ActionForm e a popolarne gli attributi con i valori presenti nella request http, facendo una corrispondenza tra nome parametro e nome attributo;

3. Se nell'elemento <action> è presente l'attributo validate al valore true chiama il metodo *validate()* dell'ActionForm per il controllo dei dati forniti dalla request;

4. Se il controllo è ok a questo punto il RequestProcessor chiama il metodo *execute()* dell'Action delegandole l'elaborazione della richiesta;

5. Il metodo *execute()* dell'Action al termine dell'elaborazione restituisce un oggetto ActionForward che consente al RequestProcessor di inoltrare il flusso elaborativi all'oggetto successivo, sia essa una pagine JSP o un'altra Action.

La classe *org.apache.struts.Action* è l'elemento fondamentale del controller di Struts, essa contiene al proprio interno il metodo *execute()*, al cui interno lo sviluppatore inserisce il proprio codice di elaborazione della richiesta per la funzione specifica.
Gli step principali eseguiti da una Action sono:

1. Acquisire i dati della request dal form;

2. Delegare l'elaborazione della business logic alle classi del Model;

3. Acquisire i risultati dell'elaborazione e prepararli per la vista da inviare all'utente mettendoli nello scope opportuno (se necessario);

4. Inoltrare il flusso elaborativo in base alla logica applicativa.

In pratica la Action è il "ponte applicativo" tra il Controller ed il Model di una applicazione Struts.

Altra classe molto importante è la *org.apache.struts.action.ActionForm*. I form, che estendono tale classe, sono, sostanzialmente, dei bean contenenti dati, che il framework popola con i dati della request, liberando di tale compito lo sviluppatore. Associando il form ad una Action, viene data la possibilità al metodo *execute()* di tale Action di avere a disposizione tutti i dati inseriti in un Form Html, con la possibilità di validarli prima che gli stessi giungano alla Action stessa tramite il metodo *validate()*; è come avere un "firewall" che "protegge" lo strato di logica dai dati non validi.

- **Componenti della View**

La view ha due compiti fondamentali: presentare all'utente i risultati di una elaborazione eseguita e consentire all'utente l'immissione di dati da elaborare.

Per quanto riguarda la presentazione di dati all'utente si utilizzano i tag JSP per visualizzare i contenuti di JavaBeans oppure altri oggetti. Invece, per consentire l'immissione di dati all'utente si realizza un apposito Form HTML nella pagina JSP e si estende la classe ActionForm per validarli ed acquisirli.

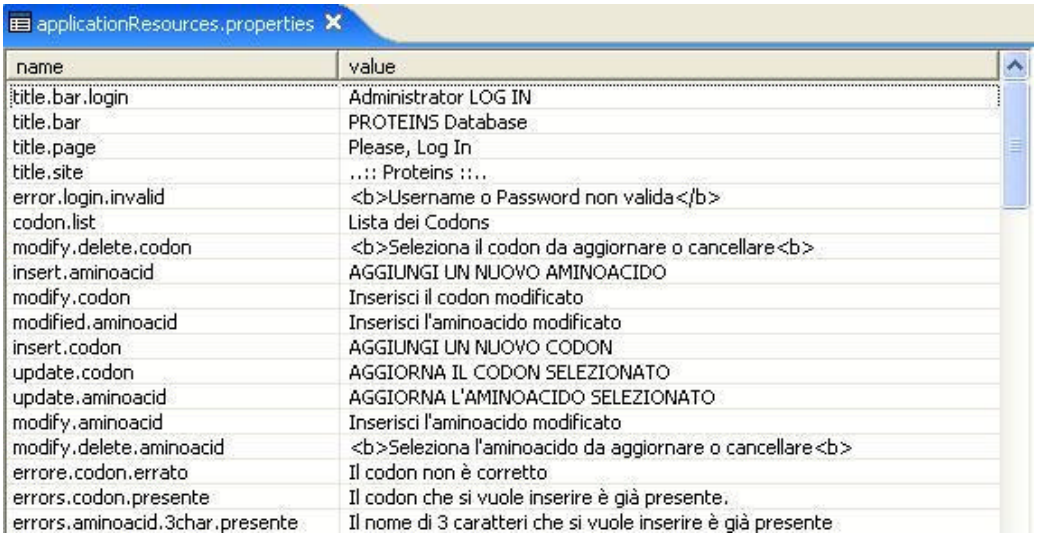
Per la costruzione delle view delle applicazioni, Struts mette a disposizione alcune librerie di tag, che svolgono i compiti più frequenti. Tra queste le più utilizzate sono:

- Html tag, per la generazione degli elementi html;
- Bean tag, per accedere a proprietà di JavaBeans e per creare istanze di bean;
- Logic tag, per implementare logica condizionale, iterazioni e controllo di flusso.

In aggiunta a queste librerie specifiche di Struts è possibile utilizzare la JSP Standard Tag Library (JSTL), nata per permettere agli sviluppatori di trasferire la proprie pagine JSP da un container ad un altro senza modificare le pagine stesse. Tale libreria, allo stato attuale, non permette di eliminare le tre librerie viste prima, ma solo alcuni tag.

L'utilizzo dei custom-tag è altamente consigliato, in quanto essi non prevedono l'introduzione di logica all'interno delle pagine, a differenza degli scriptlet. Gli sviluppatori suggeriscono di utilizzare esclusivamente tag, al fine di rispettare il più possibile il design pattern MVC. Ovviamente si è seguita tale direttiva nella realizzazione dell'applicazione in esame.

Altro aspetto importante della view è rappresentato dalla possibilità di raggruppare tutti i messaggi in un resource bundle(Fig. 3.2), con estensione *.properties*, ottenuto associando il messaggio ad un identificativo unico. Così facendo all'interno delle pagine JSP si troveranno solo gli identificativi dei messaggi. Questa possibilità comporta notevoli vantaggi in termini di manutenzione dell'applicazione, in quanto se dovesse sorgere in futuro la necessità di modificare un messaggio presente in più pagine esso dovrà essere cambiato solo nel file delle proprietà e non in ogni pagina, e in termini di internazionalizzazione, come si vedrà in seguito.


applicationResources.properties

name	value
title.bar.login	Administrator LOG IN
title.bar	PROTEINS Database
title.page	Please, Log In
title.site	..:: Proteins ::..
error.login.invalid	Username o Password non valida
codon.list	Lista dei Codons
modify.delete.codon	Seleziona il codon da aggiornare o cancellare
insert.aminoacid	AGGIUNGI UN NUOVO AMINOACIDO
modify.codon	Inserisci il codon modificato
modified.aminoacid	Inserisci l'aminoacido modificato
insert.codon	AGGIUNGI UN NUOVO CODON
update.codon	AGGIORNA IL CODON SELEZIONATO
update.aminoacid	AGGIORNA L'AMINOACIDO SELEZIONATO
modify.aminoacid	Inserisci l'aminoacido modificato
modify.delete.aminoacid	Seleziona l'aminoacido da aggiornare o cancellare
errore.codon.errato	Il codon non è corretto
errors.codon.presente	Il codon che si vuole inserire è già presente.
errors.aminoacid.3char.presente	Il nome di 3 caratteri che si vuole inserire è già presente

Figura 3.2 - Estratto del file ApplicationResource.properties

3.4 Caratteristiche di Base di Struts

E' ora possibile individuare alcune caratteristiche peculiari di Struts, che sono comuni anche ad altri MVC framework.

Esiste una sola servlet di controllo centralizzata. Tutte le richieste sono mappate sulla ActionServlet nel file *web.xml* dell'applicazione. Ciò consente di avere un unico punto di gestione del flusso applicativo e quindi permette di implementare in modo univoco e centralizzato funzioni quali sicurezza, logging, etc.

Le viste dell'applicazione non contengono al loro interno il riferimento al flusso dell'applicazione e non contengono logica applicativa. I lvelli logici dell'applicazione sono disaccoppiati;

Le viste sono identificate con nomi logici definiti nel file di configurazione struts-config.xml. Nel codice Java non è presente alcun riferimento a nomi di pagine JSP, il che rende molto più semplice variare il flusso applicativo.

Tutta la configurazione dell'applicazione è scritta esternamente in un file XML il che consente di modificare le associazioni tra le richieste http e le classi ad essa associate in modo molto semplice.

3.5 Principali Vantaggi nell'Uso di Struts

Da quanto esposto è già possibile evidenziare alcune delle caratteristiche di una applicazione sviluppata con Struts e alcuni vantaggi conseguenti al suo utilizzo:

- *Modularità e Riusabilità* : I diversi ruoli dell'applicazione sono affidati a diversi componenti. Ciò consente di sviluppare codice modulare e più facilmente riutilizzabile;

- *Manutenibilità*: L'applicazione è costituita da livelli logici ben distinti. Una modifica in uno dei livelli non comporta modifiche negli altri. Ad esempio una modifica ad una pagina JSP non ha impatto sulla logica di controllo o sulla logica di business, cosa che avviene nel JSP Model 1;

- *Rapidità di sviluppo*: A differenza di quanto avviene utilizzando il JSP Model 1, è possibile sviluppare in parallelo le varie parti dell'applicazione, view (JSP/HTML) e logica applicativa (Java), sfruttando al meglio le conoscenze dei componenti del team di sviluppo. Si possono utilizzare sviluppatori meno esperti e anche con poche conoscenze di Java per la realizzazione delle view, permettendo agli sviluppatori Java più esperti di concentrarsi sulla realizzazione della logica applicativa.

Alcuni dei vantaggi suddetti, ed ulteriori notevoli vantaggi del framework Struts e di progetti ad esso collegati verranno messi in evidenza in seguito quando verranno trattati gli aspetti salienti della realizzazione dell’applicazione web.

CAPITOLO QUATTRO

PROGETTAZIONE DELL'APPLICAZIONE WEB

Questo capitolo illustra la fase di progettazione dell'applicazione Web realizzata. Questa è la fase più critica di tutto il ciclo di sviluppo software. Una progettazione accurata consente di evitare molti errori e quindi ridurre i tempi di sviluppo.

4.1 Note Iniziali

Prima di entrare nel dettaglio della progettazione dell'applicazione, si presentano le principali informazioni derivanti dallo studio del dominio applicativo, le quali saranno molto utili ai fini della progettazione stessa e della successiva fase di realizzazione.

Una proteina è costituita da più catene di aminoacidi [11], la sequenza con la quale gli aminoacidi si susseguono nelle catene rappresenta la struttura primaria della proteina. Queste catene si avvolgono o si dispiegano secondo una specifica forma, cosiddetta struttura secondaria della proteina. La composizione della struttura primaria e la forma della struttura secondaria risultano essenziali per verificare il corretto funzionamento della proteina stessa. In seguito al malfunzionamento di qualche enzima possono avvenire delle mutazioni nella sequenza degli aminoacidi, ovvero un aminoacido in una determinata posizione della catena si trasforma in un altro, modificando la struttura primaria e la struttura secondaria della proteina, comportando la modifica dei caratteri chimici della proteina stessa. Da ciò derivano tutte le conseguenze e le disfunzioni di cui si è discusso in precedenza.

Le informazioni a nostra disposizione [8] [10] [46] per la progettazione dell'applicazione sono relative ad una particolare proteina, la proteina GALT, alle relative mutazioni collegate alla malattia metabolica "galattosemia classica", e ai risultati di analisi effettuate su di essa.

La disponibilità di informazioni su di una singola proteina non impedisce di poter realizzare una applicazione generale, adatta cioè a contenere e rappresentare

informazioni riguardanti tutte le proteine, collegate a malattie metaboliche ereditarie per cui ci possa essere un particolare interesse, dato che le informazioni e i dati che andranno gestiti e quindi dovranno essere consultabili sono ottenibili per qualunque proteina.

Si anticipa che, in questo capitolo, non verranno presentati tutti i diagrammi realizzati, ma ci si limiterà a effettuare una descrizione del lavoro svolto e di un numero limitato di tali diagrammi. L'intera gamma di documenti prodotti in questa fase sono stati inseriti nell'Appendice B.

4.2 Use Cases

Gli Use Cases(o Casi d'Uso) rappresentano uno strumento utile per catturare il funzionamento esterno del sistema da sviluppare, senza dover specificare come tale comportamento viene realizzato. In pratica specificano le operazioni, tramite le quali, gli utenti dell'applicazione possono interagire con il sistema.

Essi possono essere rappresentati in due modi, con una modalità testuale e/o con una modalità grafica. Nella realizzazione di prodotti software, e quindi anche nella applicazione oggetto della tesi, è buona norma utilizzare entrambe le rappresentazioni.

Il primo passo per la costruzione dei Casi d'Uso è rappresentato dall'individuazione degli attori, ovvero degli utilizzatori del sistema. Gli attori che utilizzeranno l'applicazione sono classificabili in due tipi:

- l'Amministratore del Sistema

- l'Utente generico

Figura 4.1 - Attori della applicazione

In realtà l'Amministratore del Sistema rappresenta una specializzazione dell'Utente generico, come si può notare dalla figura 4.1, in quanto è un Utente generico con delle funzionalità aggiuntive.

Per entrambi gli utenti non sono richieste particolari conoscenze informatiche tranne la familiarità con gli ambienti a finestre e la capacità di utilizzo di un browser Web.

Per semplicità di rappresentazione dei risultati della fase progettuale si ritiene utile effettuare una suddivisione in due moduli, i quali non presentano aspetti comuni e pertanto sarebbe stato possibile anche progettarli separatamente in parallelo.
Tali moduli sono così caratterizzati:

- Modulo Amministrazione, contenente le funzionalità a cui può accedere esclusivamente l'Amministratore, in cui vengono effettuate le operazioni di gestione dei dati e delle informazioni relative all'amministratore;

- Modulo pubblico, contenente le funzionalità accessibili a tutti gli utenti, quale, ad esempio, la consultazione delle informazioni sulle proteine.

- **Modulo Amministrazione**

Viene riportato il diagramma dei casi d'uso (Fig. 4.2):

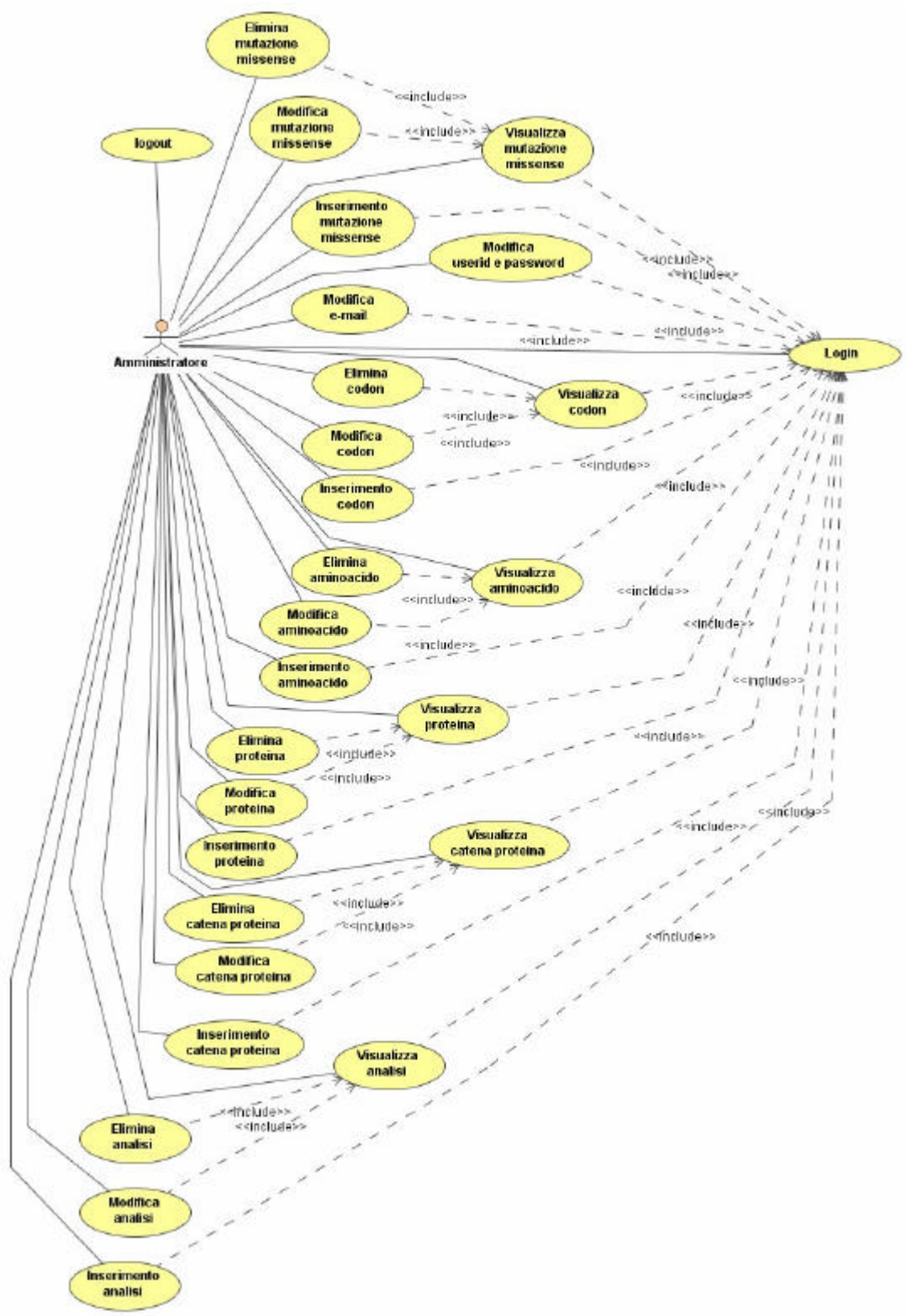


Figura 4.2 - Diagramma dei casi d'uso per il Modulo Amministrazione

Il diagramma illustra in maniera chiara e sintetica tutte le operazioni con cui l'Utente Amministratore può interagire con il Sistema, dal punto di vista dell'utente e non delle funzionalità interne al Sistema.

Come è possibile rilevare dal diagramma il modulo amministrazione comprende tutte le operazioni inerenti all'inserimento, all'aggiornamento, all'eliminazione e alla visualizzazione delle informazioni inerenti alla proteina, oltre alle operazioni di gestione delle informazioni relative all'amministratore e al Login. Tutti i casi d'uso comprendono una relazione di inclusione del caso d'uso Login. Ciò significa che per poter portare a termine una qualunque delle operazioni è necessario che sia stato effettuato il Login. Per semplicità di rappresentazione non sono stati rappresentati i casi d'uso relativi a tutte le analisi possibili, ma solo ad una generica analisi.

Per capire meglio il significato di ogni caso d'uso sarebbe utile riportare la descrizione testuale di ciascuno di essi, ma ciò rappresenta un appesantimento ritenuto inutile e quindi ci si limiterà a presentarne solo alcuni. In particolare si è deciso di presentare i casi d'uso relativi alla gestione dei dati sui nomi delle proteine, rimandando all'Appendice B per gli altri. Questa scelta è del tutto arbitraria e non derivante da motivazioni particolari.

Nelle pagine seguenti vengono riportati i casi d'uso in forma testuale relativi a:

- Inserimento Proteina

- Visualizza Proteine

- Modifica Proteina

- Elimina Proteina

Un caso d'uso viene descritto sotto forma di "Scenario di interazione" tra l'utilizzatore e il sistema. E' possibile individuare uno o più Scenari, uno denominato "normale" che rappresenta ciò che accade quando non si presentano intoppi nello svolgimento dell'interazione, gli altri denominati "alternativi" quando nello svolgimento dell'interazione interviene qualche evento eccezionale, come, ad esempio, il mancato inserimento dei dati obbligatori, che modifica la regolare esecuzione del caso d'uso.

Caso d'uso: Inserimento Proteina

Scenario 1 (Scenario Normale)

1. **include Login**
2. L'amministratore inserisce i dati relativi a proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che la proteina non sia già presente nella banca dati
5. Il sistema registra i dati nella banca dati

Scenario 2 (Scenario Alternativo)

1. **include Login**
2. L'amministratore inserisce i dati relativi a proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema informa l'amministratore i dati obbligatori non sono stati inseriti
5. L'amministratore inserisce i dati obbligatori o annulla l'inserimento

Scenario 3 (Scenario Alternativo)

1. **include Login**
2. L'amministratore inserisce i dati relativi a proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che la proteina non sia già presente nella banca dati
5. Il sistema informa l'amministratore che la proteina è già presente nella banca dati

Caso d'uso: Visualizza proteine

Scenario 1 (Scenario Normale)

1. **include Login**

2. L'amministratore richiede l'elenco delle proteine presenti nella banca dati
3. Il sistema verifica che siano presenti proteine nella banca dati
4. Il sistema fornisce l'elenco delle proteine

Scenario 2 (Scenario Alternativo)

1. **include Login**
2. L'amministratore richiede l'elenco delle proteine presenti nella banca dati
3. Il sistema verifica che siano presenti proteine nella banca dati
4. Il sistema informa l'Amministratore che non sono presenti proteine nella banca dati
5. L'Amministratore annulla l'operazione

Caso d'uso: Modifica proteina

Scenario 1 (Scenario Normale)

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da modificare
3. Il sistema verifica se il codice inserito è valido
4. L'amministratore modifica i dati relativi alla proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che la proteina modificata non sia già presente nella banca dati
7. Il sistema verifica se la proteina modificata è presente in qualche Mutazione Missense o in qualche catena della proteina e in caso affermativo provvede all'aggiornamento in cascata
8. Il sistema provvede a registrare i nuovi dati

Scenario 2 (Scenario Alternativo)

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da modificare
3. Il sistema verifica se il codice inserito è valido
4. Il sistema informa l'amministratore che il codice inserito non è valido
5. L'amministratore modifica il codice o annulla le modifiche

Scenario 3 (Scenario Alternativo)

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da modificare
3. Il sistema verifica se il codice inserito è valido
4. L'amministratore modifica i dati relativi alla proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
7. L'amministratore inserisci i dati obbligatori o annulla le modifiche

Scenario 4 (Scenario Alternativo)

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da modificare
3. Il sistema verifica se il codice inserito è valido
4. L'amministratore modifica i dati relativi alla proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che la proteina modificata non sia già presente nella banca dati

7. Il sistema informa l'amministratore che la proteina è già presente nella banca dati
8. L'amministratore modifica il codice della proteina o annulla le modifiche

Caso d'uso: Elimina proteina

Scenario 1 (Scenario Normale)

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da eliminare
3. Il sistema verifica se il codice inserito è valido
4. Il sistema verifica se alla proteina è già associato almeno un elemento nella catena della proteina o a qualche mutazione missense
5. Il sistema elimina la proteina

Scenario 2 (Scenario Alternativo)

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da eliminare
3. Il sistema verifica se il codice inserito è valido
4. Il sistema informa l'amministratore che il codice inserito non è valido
5. L'amministratore modifica il codice o annulla l'eliminazione

Scenario 3 *(Scenario Alternativo)*

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da eliminare
3. Il sistema verifica se il codice inserito è valido
4. Il sistema verifica se alla proteina è già associato almeno un elemento nella catena della proteina o a qualche mutazione missense
5. Il sistema informa l'amministratore che la proteina non è eliminabile in quanto c'è qualche elemento nella catena della proteina e/o qualche mutazione missense associato ad essa
6. L'amministratore modifica il codice da eliminare o annulla l'eliminazione

E' importante far notare la grande utilità della relazione di inclusione, in quanto, se non fosse possibile effettuare tale rappresentazione, bisognerebbe ripetere, ad esempio, per ogni scenario del caso d'uso Modifica Proteina, gli scenari del caso d'uso Visualizza Proteine, che a sua volta include il caso d'uso Login con tutti gli Scenari possibili. Ciò comporterebbe una gran confusione e una leggibilità pessima.

- **Modulo Pubblico**

Si riporta il diagramma dei casi d'Uso relativo a tale modulo (Fig. 4.3):

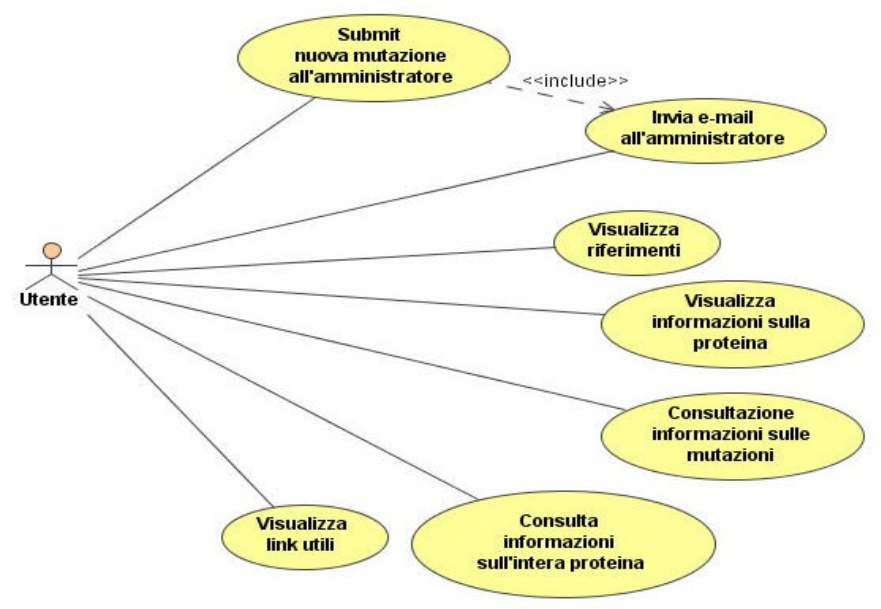


Figura 4.3 - Diagramma dei Casi d'Uso per il Modulo Pubblico

All'interno del modulo, definito Pubblico, sono presenti tutte le operazioni rese disponibili ad un generico utente che si collega tramite un browser Web all'applicazione. Ovviamente tale diagramma è relativo ad un'unica proteina generica, non è necessario realizzare un diagramma diverso per ogni proteina, in quanto le operazioni che si possono effettuare sono uguali per tutte. Come si può notare è possibile visualizzare informazioni sulla proteina, link a siti informativi, riferimenti. Inoltre è possibile consultare informazioni specifiche sulle proteine e sulle relative mutazioni, oltre a dare la possibilità agli utenti di contattare l'amministratore, tramite e-mail, per fornire dati su nuove mutazioni eventualmente individuate.

Relativamente a tale modulo, si riporta il caso d'uso in forma testuale:

Caso d'uso: Submit nuova mutazione all'Amministratore

Scenario 1 (Scenario Normale)

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema verifica che Aminoacido Mutante e Mutato non coincidano
5. Il sistema verifica che Codon Mutante e Mutato non coincidano
6. Il sistema verifica l'esistenza della proteina nella banca dati
7. Il sistema verifica che il residuo e l'Aminoacido Mutante corrispondano ad un elemento della catena della proteina
8. **Include Invio e-mail all'Amministratore**

Scenario 2 (Scenario Alternativo)

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema informa l'utente che i dati obbligatori non sono stati inseriti
4. L'utente inserisce i dati obbligatori o annulla la sottomissione

Scenario 3 (Scenario Alternativo)

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema informa l'utente che l'indirizzo E-mail inserito non è corretto
5. L'utente modifica i dati o annulla la sottomissione

Scenario 4 (Scenario Alternativo)

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema verifica che Aminoacido Mutante e Mutato non coincidano
5. Il sistema informa l'utente che gli Aminoacidi inseriti coincidono
6. L'utente modifica i dati o annulla la sottomissione

Scenario 5 (Scenario Alternativo)

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema verifica che Aminoacido Mutante e Mutato non coincidano
5. Il sistema verifica che Codon Mutante e Mutato non coincidano
6. Il sistema informa l'utente che i Codon inseriti coincidono
7. L'utente modifica i dati o annulla la sottomissione

Scenario 6 (Scenario Alternativo)

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema verifica che Aminoacido Mutante e Mutato non coincidano
5. Il sistema verifica che Codon Mutante e Mutato non coincidano
6. Il sistema verifica l'esistenza della proteina nella banca dati
7. Il sistema informa l'utente che la proteina inserita non esiste nella banca dati
8. L'utente modifica i dati o annulla la sottomissione

Scenario 7 (Scenario Alternativo)

4.1 L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
4.2 Il sistema verifica che i dati obbligatori siano stati inseriti
4.3 Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4.4 Il sistema verifica che Aminoacido Mutante e Mutato non coincidano
4.5 Il sistema verifica che Codon Mutante e Mutato non coincidano
4.6 Il sistema verifica l'esistenza della proteina nella banca dati
4.7 Il sistema verifica che il residuo e l'Aminoacido Mutante corrispondano ad un elemento della catena della proteina
4.8 Il sistema informa l'utente che il residuo e l'Aminoacido Mutante non corrispondono a nessun elemento della catena della proteina
4.9 L'utente modifica l'e-mail o annulla la sottomissione

4.3 Costruzione del Modello Concettuale

4.3.1 Modello Concettuale

Riportiamo le informazioni necessarie alla costruzione del modello concettuale: una proteina è caratterizzata da un nome e da un codice, la sua struttura primaria è formata da una sequenza di aminoacidi ed è chiamata catena della proteina. Ogni aminoacido è codificato mediante un codon, ovvero una tripletta di nucleotidi. Per ogni aminoacido presente nella catena è presente un insieme di analisi, che verranno brevemente descritte nel dizionario dei dati. Infine si ha a disposizione l'insieme delle Mutazioni Missense conosciute fino ad oggi, compreso il codon mutante. Ovviamente abbiamo anche a disposizione i dati relativi all'Amministratore.

E' ora possibile costruire il modello concettuale, o modello E-R, realizzato con il CASE PowerDesigner (Fig. 4.4):

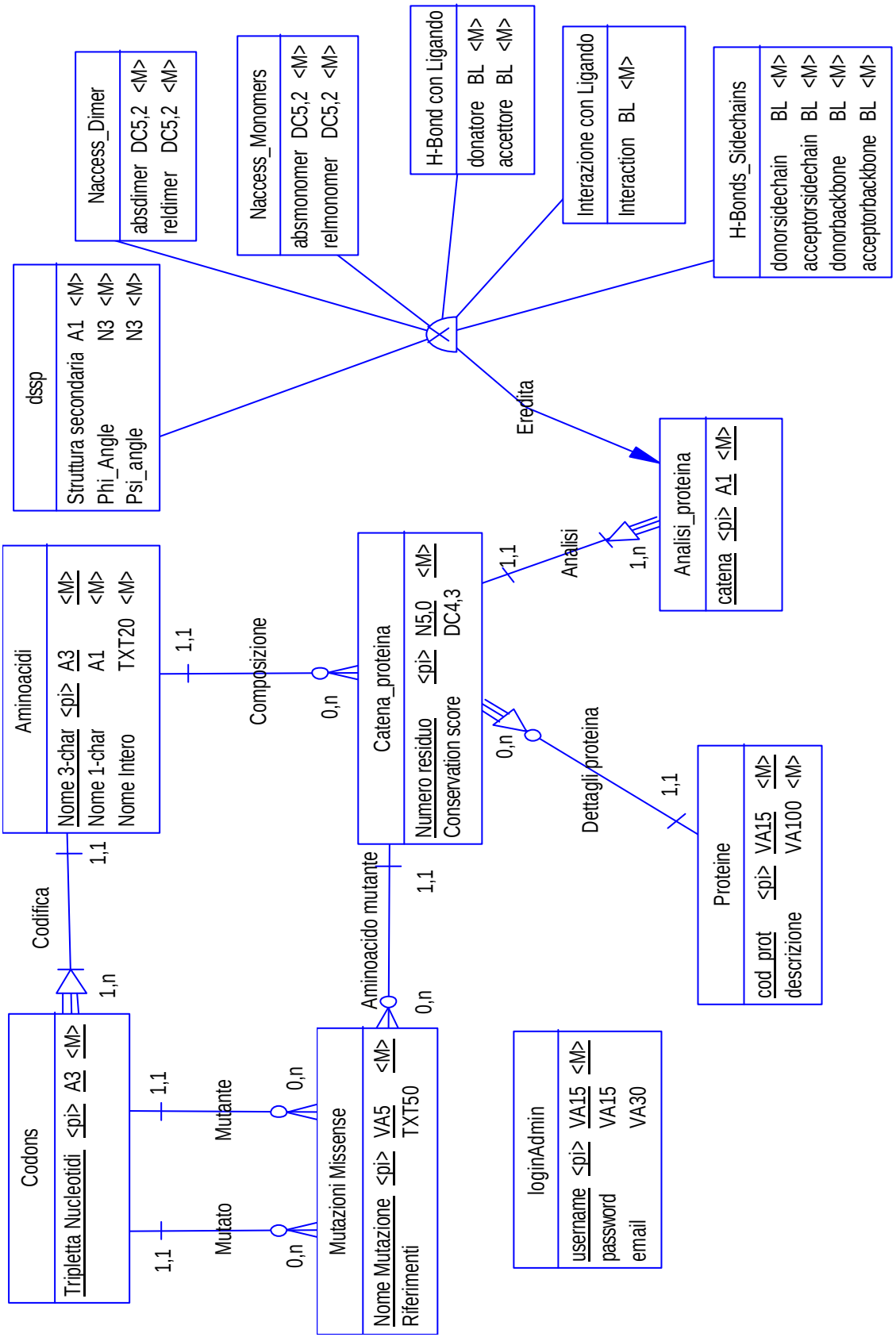


Figura 4.4 - Modello concettuale dell'applicazione

4.3.2 Dizionario dei Dati

Presentiamo ora delle piccole schede delle singole entità. Per ognuna si dà una breve descrizione e, laddove necessario, delle spiegazioni aggiuntive sugli attributi. Per avere descrizioni più specifiche si rimanda a testi tecnici [8] [10] [11].

- ***loginAdmin***

loginAdmin		
username	VARCHAR(15)	<pk>
password	VARCHAR(15)	
email	VARCHAR(30)	

Contiene tutte le informazioni relative all'amministratore del sistema, al fine di autenticare colui che dovrà gestire i dati e di poter inviare informazioni per e-mail.

- ***Codons***

Codons			
Tripletta Nucleotidi	<pi>	A3	<M>

Contiene tutte le possibili combinazioni di triplette di nucleotidi che possono codificare gli aminoacidi.

- ***AminoAcidi***

Aminoacidi			
Nome 3-char	<pi>	A3	<M>
Nome 1-char		A1	<M>
Nome Intero		TXT20	<M>

Contiene i nomi e le rappresentazioni dei 20 aminoacidi esistenti. In particolare i vari attributi rappresentano:

- Nome intero: è il nome esteso dell'aminoacido(Es. Alanine)
- Nome 3-char e Nome 1-char: ogni aminoacido viene identificato con un codice di 3 caratteri e con un codice di 1 carattere. Il più utilizzato, o meglio quello più spesso trovato nelle nostre ricerche è quello di 3 caratteri, ma si è preferito riportarli entrambi.(Es. Ala o A)

- ***Proteine***

Proteine			
cod_prot	<pi>	VA15	<M>
descrizione		VA100	<M>

Ogni proteina ha un proprio nome tecnico, individuato con l'attributo "descrizione", di lunghezza molto elevata. Per tale motivo le proteine vengono identificate con un codice abbreviato.

- ***Catena Proteina***

Catena_proteina			
Numero residuo	<pi>	N5,0	<M>
Conservation score		DC4,3	

La catena della proteina, come si è già detto, rappresenta la struttura primaria della proteina, ovvero la sequenza degli aminoacidi. Ogni elemento della catena viene anche individuato come "residuo". Ciascun elemento della catena della proteina è caratterizzato da:

- o Numero di residuo, numero crescente che caratterizza l'ordine degli aminoacidi all'interno della catena;
- o Conservation score, o grado di conservazione del residuo, che rappresenta il grado di conservazione dell'elemento, all'interno della catena della proteina, in specie differenti dall'uomo.

- ***Mutazioni Missense***

Mutazioni Missense			
Nome Mutazione	<pi>	VA5	<M>
Riferimenti		TXT50	

Contiene le informazioni relative alle mutazioni, in particolare il Nome Mutazione è un attributo ridondante in quanto ottenuto unendo le seguenti informazioni:

- nome ad 1 carattere dell'aminoacido mutante
- numero di residuo dell'elemento della catena che muta

- nome ad 1 carattere dell'aminoacido mutato

Vi sono inoltre presenti i riferimenti, ovvero il codice del testo o dell'articolo tecnico dove è stata riportata per la prima volta la mutazione.

- ***Analisi Proteina***

Analisi_proteina			
catena	<pi>	A1	<M>

Per ogni elemento della catena della proteina, come già detto, sono presenti delle analisi. Ogni proteina non è composta da una singola catena di aminoacidi(monomero), ma da due catene uguali (dimero), ciascuna identificata da una lettera maiuscola, tra di loro intrecciate in particolari forme. Le analisi sono effettuate su ogni singolo elemento di ogni catena.

- ***DSSP***

dssp		
Struttura secondaria	A1	<M>
Phi_Angle	N3	<M>
Psi_angle	N3	<M>

L'analisi con DSSP viene effettuata per individuare la struttura secondaria della proteina. In particolare:

- Struttura secondaria, rappresenta la forma assunta dalla catena. Questa forma risulta essenziale al funzionamento della proteina stessa. I possibili

valori assunti sono eliche-alfa(H), Beta-sheet(E), Turn(T) e altri derivati da questi(G,S,B,I). Quando non presente la proteina non ha una forma ben definita e viene indicata con "-";

- Phi angle e Psi angle, sono dei valori di angoli utili per individuare il particolare aminoacido all'interno della struttura secondaria.

- ***Naccess Monomers e Naccess Dimer***

Naccess_Monomers		
absmonomer	DC5,2	<M>
relmonomer	DC5,2	<M>

Naccess_Dimer		
absdimer	DC5,2	<M>
reldimer	DC5,2	<M>

L'analisi NACCESS mira ad individuare la posizione dell'aminoacido della catena della proteina espressa in termini assoluti o relativi. In particolare l'analisi sui monomeri individua la posizione dell'aminoacido all'interno o sulla superficie del singolo monomero della proteina, quella sul dimero individua la posizione all'interno o sulla superficie del dimero dell'intera proteina. In caso di valori differenti tra le analisi per i monomeri e per il dimero ciò implica che gli aminoacidi sono sulla superfici per quanto riguarda i monomeri, ma nella composizione del dimero risultano interni ad esso.

- ***H-BOND con Ligando e Interazione con Ligando***

H-Bond con Ligando		
donatore	BL	<M>
accettore	BL	<M>

Interazione con Ligando		
Interaction	BL	<M>

La prima entità contiene i risultati dell'analisi dei legami-idrogeno tra la proteina e il ligando, dove il ligando è una particolare molecola legata alla proteina. I due attributi indicano se il particolare aminoacido all'interno della catena della proteina è un donatore o un accettore di questo tipo di legame.

La seconda indica se ci sono interazioni tra l'elemento della proteina e il ligando.

- ***H-BONDS con Sidechains***

H-Bonds_Sidechains		
donorsidechain	BL	<M>
acceptorsidechain	BL	<M>
donorbackbone	BL	<M>
acceptorbackbone	BL	<M>

Prima di definire il tipo di analisi si premette che il Sidechain indica la catena laterale della proteina, mentre il Backbone rappresenta la dorsale della proteina. L'individuazione di questi due elementi dipendono dalla forma che la proteina assume. I vari attributi indicano, se l'elemento della catena, facente parte del Sidechain o del Backbone, rappresenta un accettore o un donatore di legami-idrogeno.

4.3.3 Vincoli

Si specificano ora tutti i vincoli, non rilevabili direttamente dal diagramma o comunque non specificati nella descrizione delle entità, che i valori associati agli attributi devono soddisfare.

- Per quanto riguarda l'entità codon, la codifica dei codon, ovvero la tripletta dei nucleotidi, è data da tutte le possibile combinazioni degli elementi Adenine(A), Cytosine(C), Guanine(G) e Thymine(T), quindi ci sono 64 possibili combinazioni, ciascuna codifica un aminoacido ad eccezione di 3 combinazioni che vengono indicate come aminoacidi STOP;

- Per quanto riguarda gli aminoacidi, essi sono 20 oltre a quello indicato come STOP;

- In relazione alla catena della proteina, gli elementi sono individuati dal numero di residuo crescente. In particolare la catena inizia a partire dal numero 20, in quanto l'ordine di tali elementi è noto, ma non è possibile assoggettarli ad analisi in quanto non risultano individuabili nella struttura della proteina;

- Il valore del Conservation score è un numero compreso tra 0 e 1;

- I valori degli angoli per l'analisi con DSSP sono ovviamente compresi tra -360° e +360°.

4.3.4 Analisi delle Ridondanze

Dall'analisi del modello concettuale non risultano ridondanze, ad eccezione di quella relativa al nome della mutazione, che viene derivata da altri attributi.

La decisione di mantenere o eliminare una ridondanza va presa analizzando i vantaggi e gli svantaggi in termini di accessi in memoria e di occupazione di memoria con o senza il dato ridondante.

Analizzando questa ridondanza si può notare che se essa viene eliminata, per ogni riferimento ad una mutazione si ha bisogno di tre attributi e quindi di tre accessi in memoria, mentre se essa viene mantenuta si ha bisogno di un solo accesso. Inoltre l'occupazione di memoria aggiuntiva può essere ritenuta trascurabile. Ovviamente bisogna fare molta attenzione alle operazioni di aggiornamento dei dati da cui deriva il nome mutazione durante tutte le fasi del ciclo di sviluppo dell'applicazione, onde evitare di avere dei dati inconsistenti.

4.3.5 Diagramma Logico

A partire dal modello concettuale presentato, si effettua la traduzione verso il modello logico con l'ausilio dello strumento CASE **PowerDesigner**, specificando che il DBMS che verrà utilizzato è PostgreSQL. Il risultato di tale operazione, insieme al processo per il caricamento iniziale dei dati è riportato nell'Appendice A.

Prima di effettuare la traduzione bisogna decidere come gestire la relazione di generalizzazione tra l'entità ANALISI-padre e le entità ANALISI-figlie. Le possibilità sono tre:

1) accorpamento delle entità figlie della generalizzazione nell'entità padre, con l'aggiunta di un attributo per distinguere l'analisi a cui si fa riferimento;

2) Accorpamento dell'entità padre della generalizzazione nelle entità figlie;

3) Sostituzione delle generalizzazioni con delle associazioni che legano uno a uno l'entità padre con ciascuna entità figlia.

La nostra scelta è ricaduta sull'accorpamento dell'entità padre della generalizzazione nelle entità figlie, in quanto la generalizzazione è totale, ovvero ogni occorrenza dell'entità padre è occorrenza di almeno una delle entità figlie e quindi la scelta di una delle altre due alternative risulterebbe non conveniente.

Viene riportato quindi il diagramma logico finale, che rappresenta il database della nostra applicazione Web (figura 4.5).

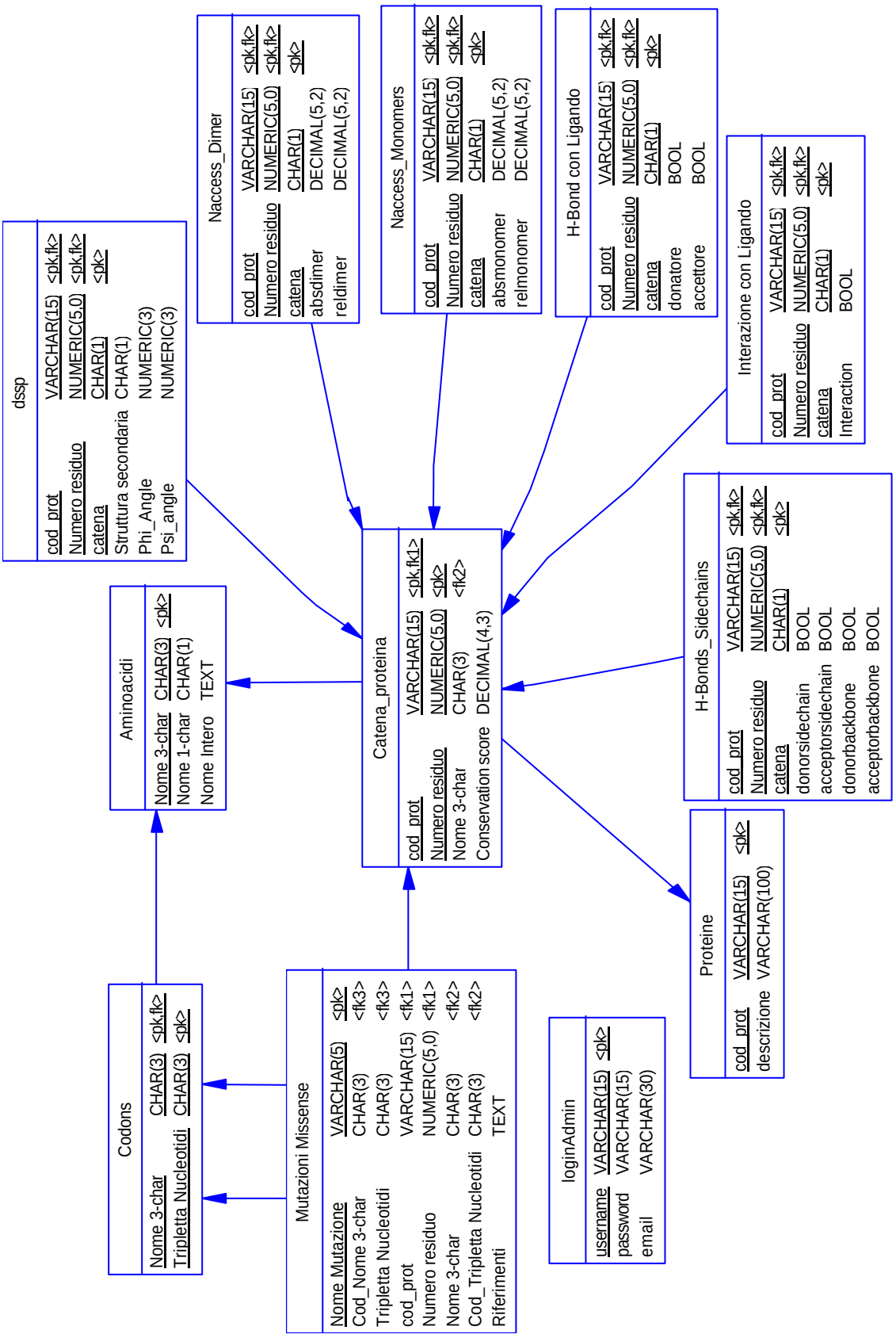


Figura 4.5 - Modello logico dell'applicazione

4.4 Class Diagrams

Il Class Diagram, o diagramma delle classi, per una applicazione Web, viene utilizzato per rappresentare l'interazione tra le varie pagine e gli eventuali altri oggetti presenti, in particolare viene creato un diagramma per ogni caso d'uso, in cui una pagina è intesa come classe del sistema.

Come per quanto visto per i Casi d'Uso, anche in questo caso ci limiteremo a rappresentare i diagrammi relativi ai casi d'Uso Inserimento Proteina(Fig. 4.6), Visualizzazione Proteina, Modifica Proteina(Fig. 4.7) ed Elimina Proteina(Fig. 4.8) per quanto riguarda il Modulo Amministrazione, mentre per il Modulo Pubblico si visualizzerà il caso d'uso Submit nuova Mutazione all'Amministratore(Fig. 4.9), rimandando all'Appendice B e al materiale allegato per tutti i diagrammi. Per ognuno di questi diagrammi si metteranno in evidenza le caratteristiche ritenute più significative.

Si precisa che, per esigenze di spazio e leggibilità dei diagrammi, le classi utilizzate per la connessione al database sono associate direttamente alle Action. In realtà le Action richiamano un Bean, con il nome uguale, ad eccezione di Bean al posto di Action, al cui interno avviene la connessione al database. Gli unici diagrammi in cui risulta presente il Bean è il caso d'uso Inserimento Proteina (sia per quanto riguarda il Class che il Sequence).

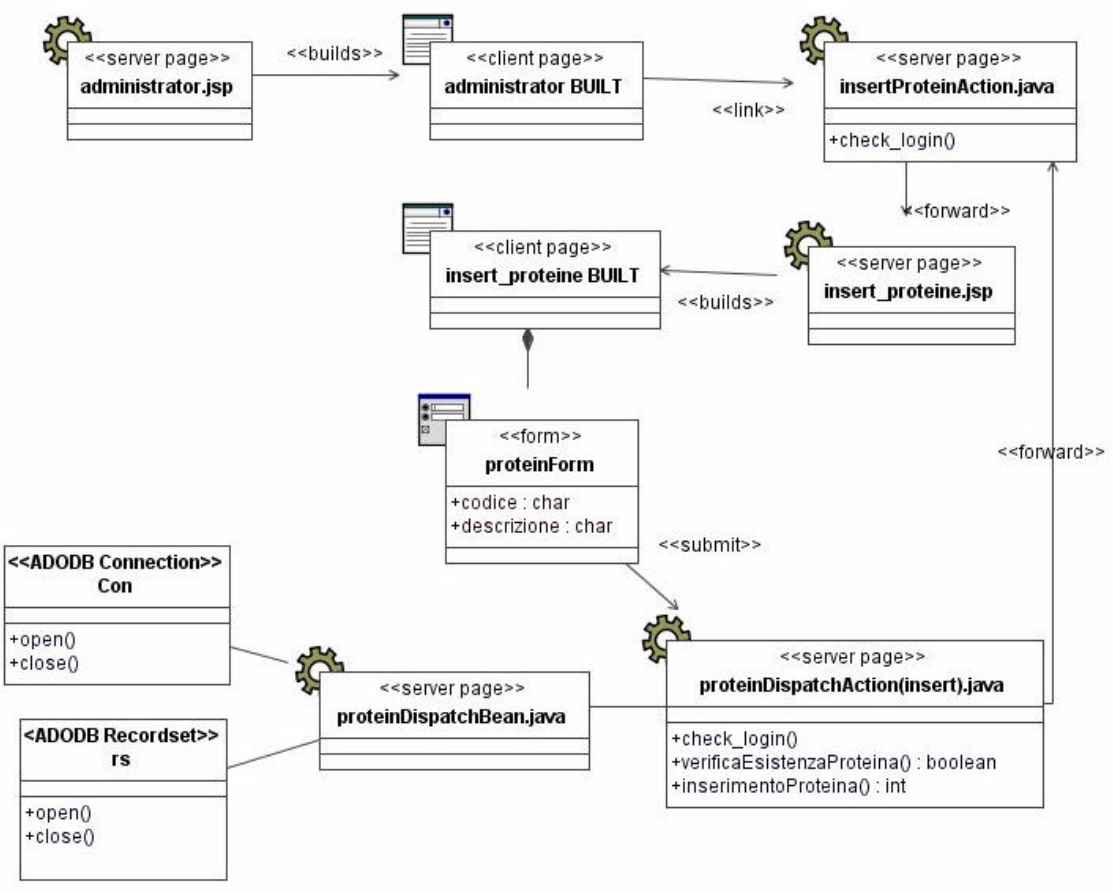


Figura 4.6 - Diagramma delle classi relativo al caso d'uso Inserimento Proteina

Come si può notare solo il form presenta attributi e solo le pagine server relative alle Action presentano dei metodi, i quali richiamano un Bean, che a sua volta si connette al database. Quest'ultima è una conseguenza della separazione tra logica di controllo e di presentazione. Analizziamo brevemente i metodi presenti:

- check_login(), serve per verificare se l'utente che accede alla pagina è o meno autorizzato, nel caso non fosse autorizzato esso viene rispedito alla pagina *loginAdmin.jsp*. Ciò non viene specificato in questo ambito in quanto questa

operazione di *redirect* andrebbe ripetuta per ogni Action. Per evitare ciò si è preferito creare un unico diagramma generico, presente in appendice;

- verificaEsistenzaProteina(), per verificare se esiste una proteina nel database con lo stesso codice inserito;

- inserimentoProteina(), effettua l'inserimento vero e proprio.

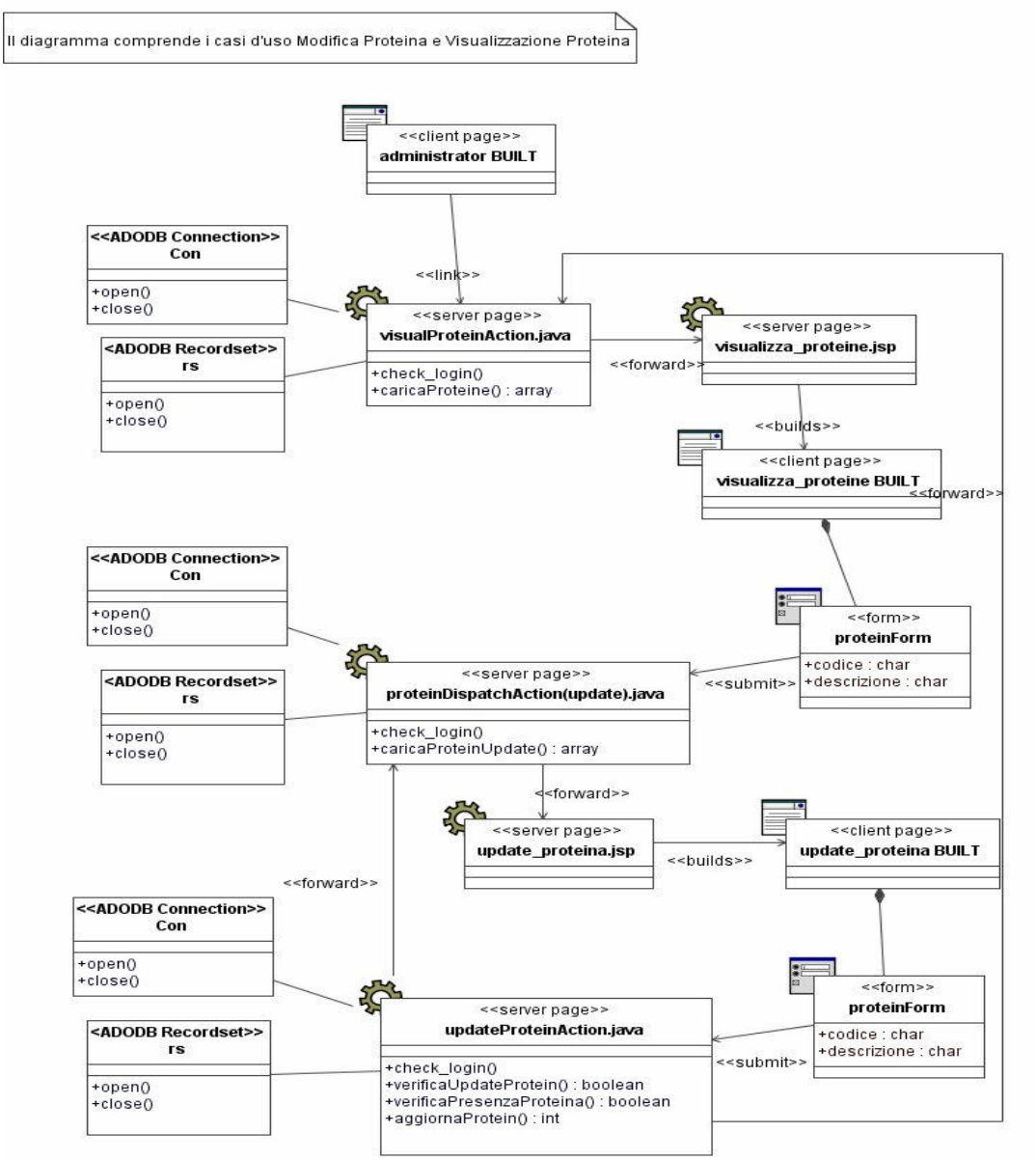


Figura 4.7 – Diagramma delle classi relativo al caso d'uso Modifica Proteina

Nel diagramma appena presentato, come in quello successivo, viene rappresentato anche il caso d'uso Visualizzazione Proteina, questo per poter rappresentare meglio i *forward* fra le pagine.

Analizziamo anche in questo caso i metodi presenti:

- check_login(), ha lo stesso funzionamento precedente;

- caricaProteine(), carica tutte le proteine presenti nel database, per farle visualizzare all'Amministratore;

- caricaProteinUpdate(), carica i dati della proteina che è stata selezionata, per permettere di modificarla;

- verificaUpdateProtein(), verifica se il codice della proteina inserito è già esistente;

- verificaPresenzaProteina(), verifica se i dati inseriti sono, per intero, già presenti, in tal caso si informa l'Amministratore che non si verranno effettuate modifiche;

- aggiornaProtein(), nel caso in cui tutti i controlli sono stati superati effettua l'aggiornamento della proteina.

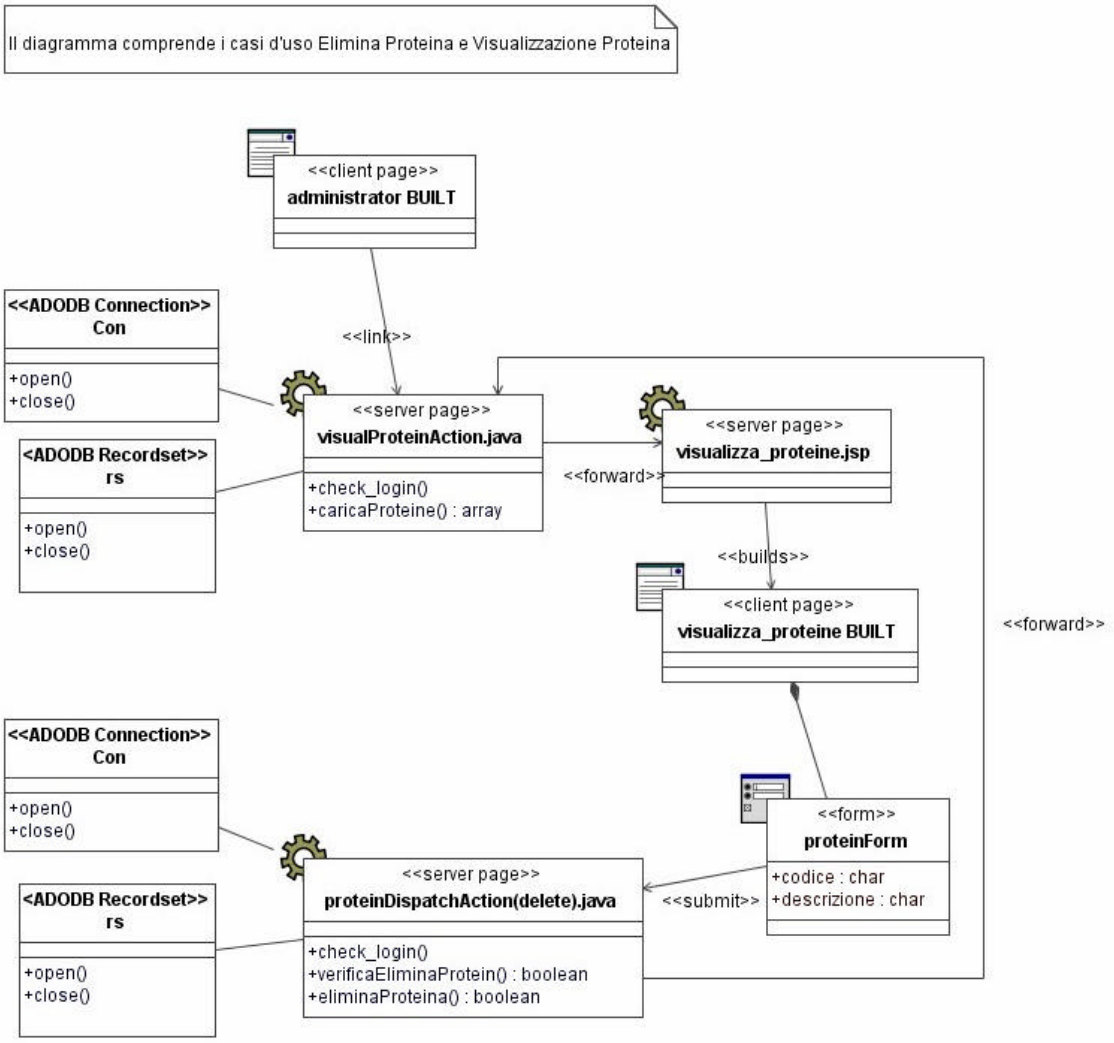


Figura 4.8 - Diagramma delle classi relativo al caso d'uso Elimina Proteina

In questo diagramma oltre ai metodi già visti prima sono presenti le funzioni:

- verificaEliminaProtein(), che serve per verificare se nella catena della proteina sono già presenti elementi associati alla proteina da eliminare, in caso affermativo non permette l'eliminazione;

- eliminaProteina(), procede alla eliminazione della proteina dal database.

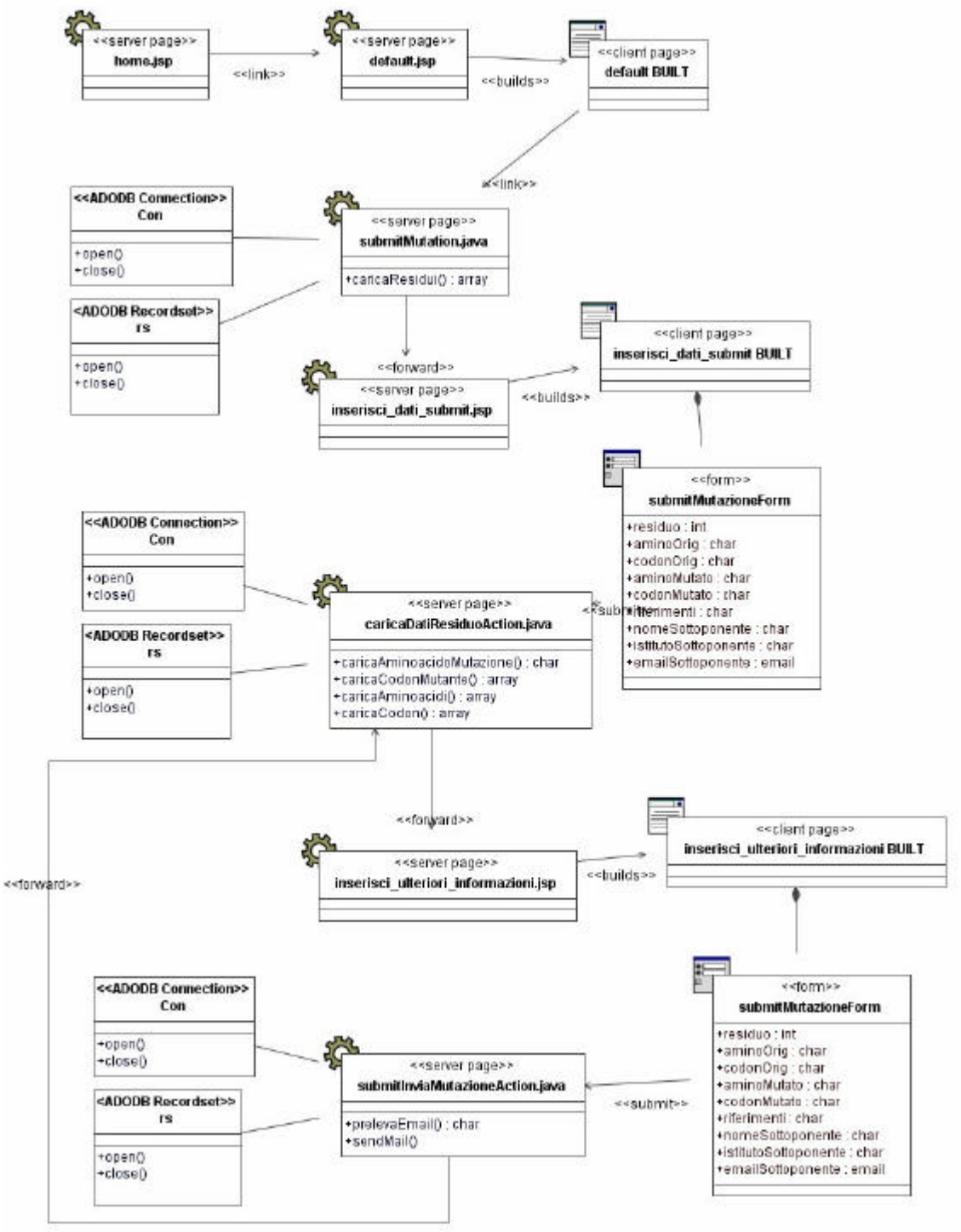


Figura 4.9 - Diagramma delle classi relativo al caso d'uso Submit nuova Mutazione

Questo è l'unico diagramma presentato del modulo Pubblico, in esso sono presenti i seguenti metodi:

- caricaResidui(), carica tutti i residui relativi alla proteina selezionata per permettere all'utente di selezionare quello relativo alla mutazione da inviare;

- caricaAminoacidoMutazione(), effettua il caricamento dell'aminoacido relativo al residuo selezionato in precedenza;

- caricaCodonMutante(), carica tutti i codon che codificano l'aminoacido mutante;

- caricaAminoacidi(), carica tutti gli aminoacidi, tra i quali andrà scelto quello corrispondente all'aminoacido mutato, ovvero quello che sostituisce il mutante in seguito alla mutazione;

- caricaCodon(), carica tutti i codon, tra i quali andrà scelto quello corrispondente al codon mutato;

- prelevaEmail(), viene prelevata l'indirizzo e-mail dell'Amministratore dal database;

- sendMail(), funzione utilizzata per l'invio dell'e-mail all'Amministratore.

Da tutti i diagrammi presentati è facile individuare le estensioni per il Web del linguaggio UML. Si effettuano distinzioni tra:

- *pagine Server*, che sono le pagine effettivamente risiedenti sul server, in cui vengono effettuate operazioni, controlli, accesso alla base dati;

- *pagine Client*, le quali sono quelle costruite e inviate al browser Web dell'utente;

- *form*, che sono contenuti nelle pagine Client, vengono creati e distrutti con esse, rappresentano il mezzo tramite il quale l'utente del sistema introduce i dati;

- le due classi *ADODB Connection* e *ADODB Recordset* che rappresentano le classi utilizzate per effettuare la connessione al database, per sottoporre query, per inserire dati, ecc.

4.5 Sequence Diagrams

Come appena visto, con i Class Diagrams è possibile individuare le interazioni tra le pagine e gli oggetti. Un altro aspetto molto importante nello sviluppo di una applicazione Web è rappresentato dallo studio delle operazioni da effettuare da un punto di vista dinamico, e non statico come avviene con i class diagram. A tal proposito si procede alla realizzazione dei Sequence Diagrams, o diagrammi di sequenza, probabilmente i diagrammi più importanti ai fini della realizzazione di una applicazione Web.

Per capire la grande importanza di tali diagrammi e mettere in evidenza la differenza con i Class Diagrams vengono riportati i Sequence Diagrams relativi ai casi d'uso visti in precedenza, ovvero Inserimento Proteina(Fig. 4.10), Visualizzazione Proteine, Modifica Proteina(Fig. 4.11), Elimina Proteina(Fig. 4.12), per il Modulo Amministrazione, e Submit nuova mutazione all'Amministratore per il modulo Pubblico(Fig. 4.13).

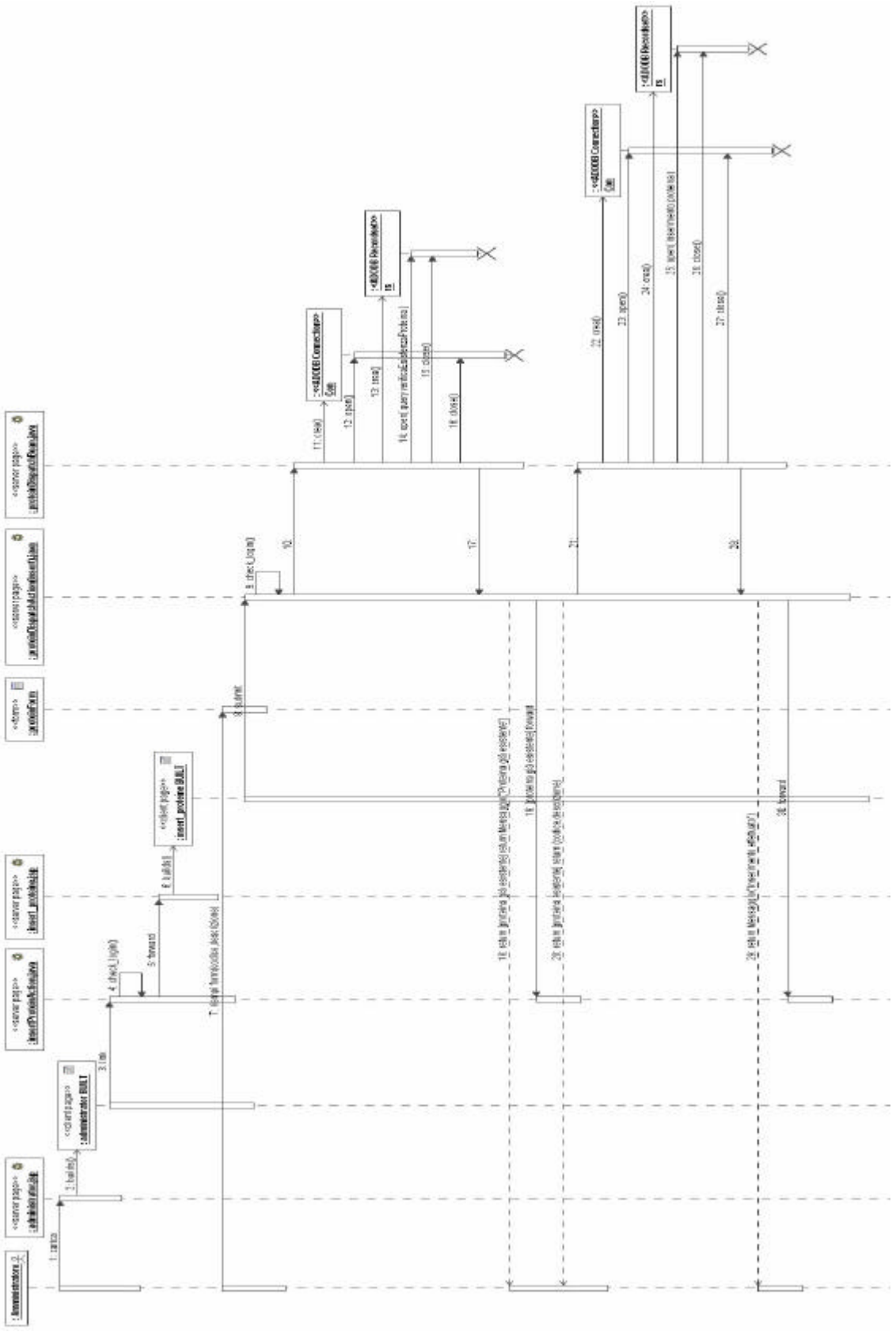


Figura 4.10 – Sequence Diagram del Caso d'Uso Inserimento proteina

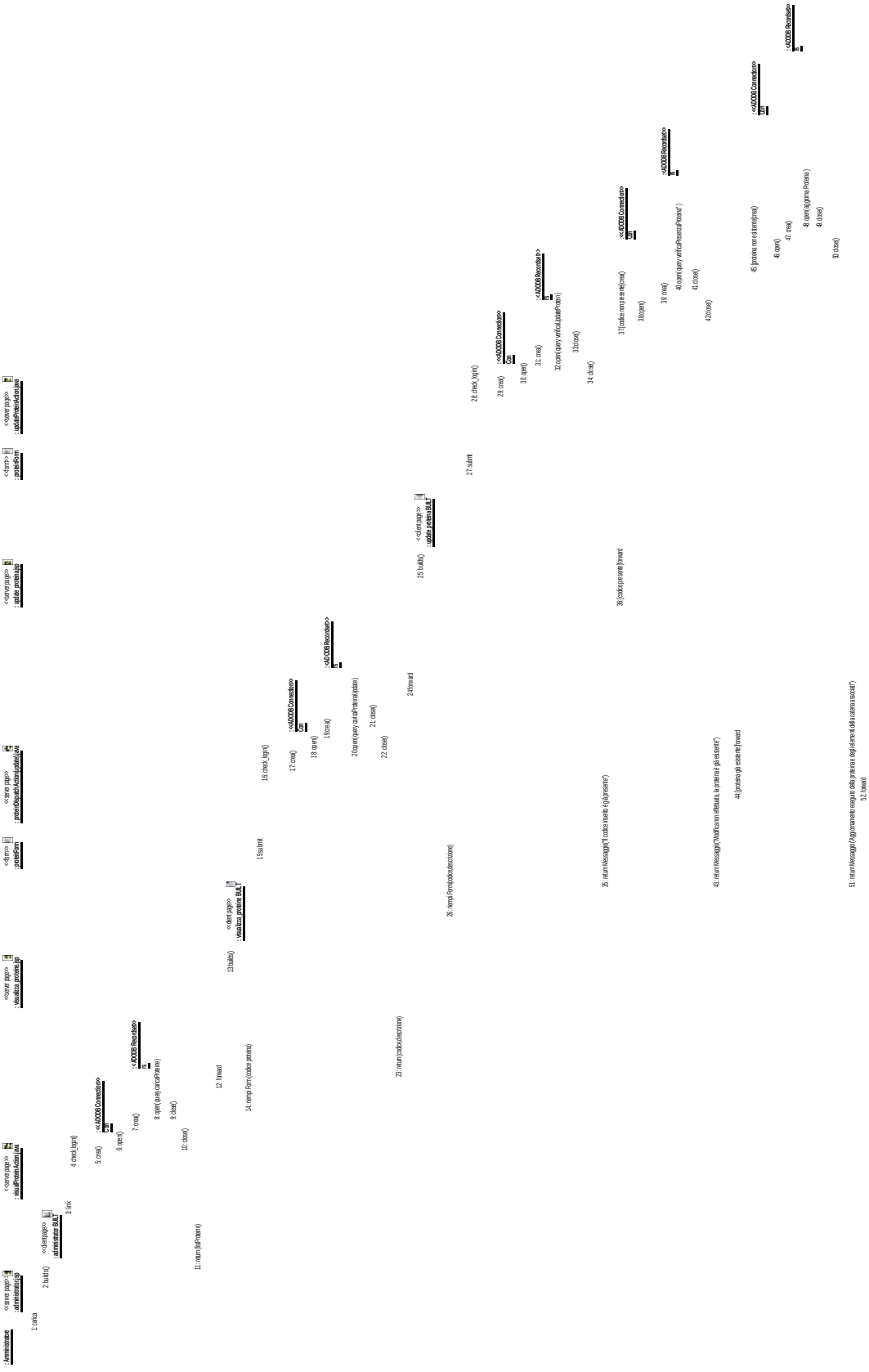


Figura 4.11 – Sequence Diagram relativo al caso d'uso Modifica Proteina

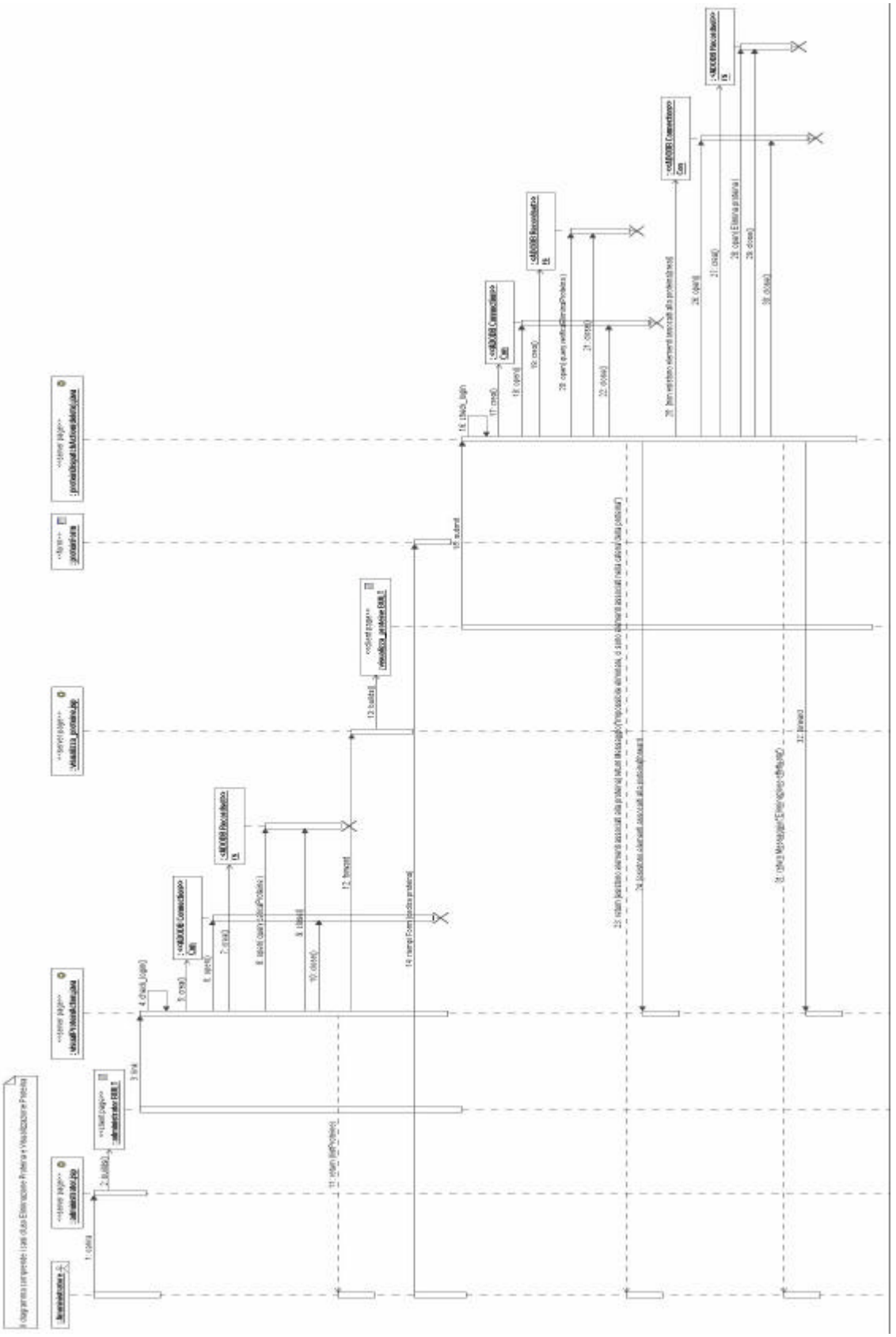


Figura 4.12 – Sequence Diagram relativo al caso d'uso Elimina Proteina

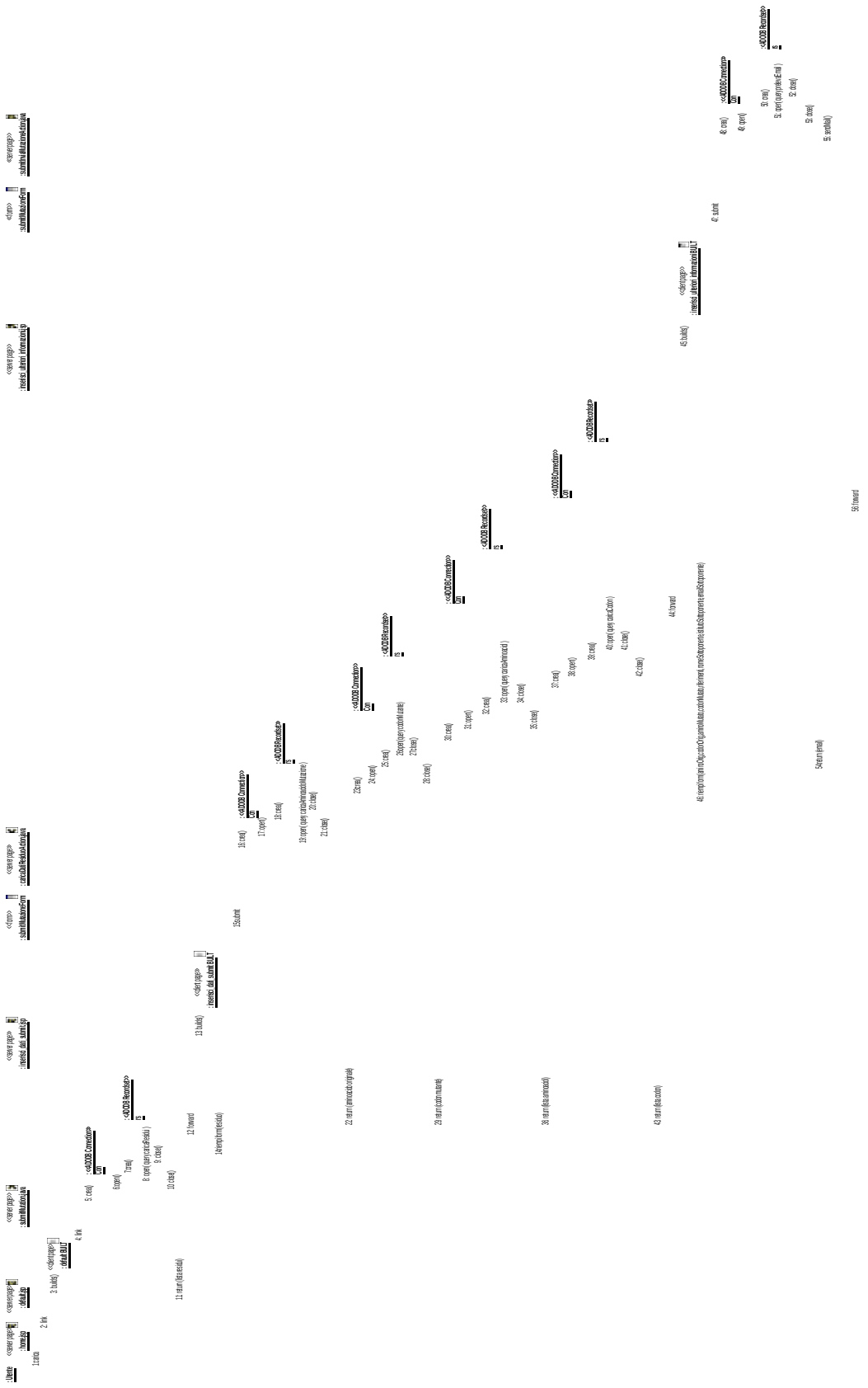


Figura 4.13 – Sequence Diagram relativo al caso d'uso Submit nuova mutazione

Come si può ben notare da tutti i diagrammi presentati, è possibile ottenere le seguenti informazioni:

- i passi che un utente deve effettuare per ottenere un determinato risultato, quale, ad esempio, l'inserimento o la modifica o l'eliminazione di una proteina da parte dell'Amministratore;

- i messaggi che vengono inviati all'utente nel corso delle interazioni e le pagine che vengono costruite e rese disponibili all'utente, in seguito ad una determinata richiesta;

- l'ordine con cui vengono effettuate le operazioni all'interno di una pagina Server. Infatti, analizzando un Class Diagram si può vedere, che in una determinata pagina sono disponibili diversi metodi, ma non è possibile, in nessun modo, capire in che ordine questi metodi vengono richiamati, cosa invece chiarissima in questi ultimi diagrammi. E' così possibile ottenere il procedimento tramite il quale la pagina Client viene costruita.

Dai diagrammi presentati è possibile individuare una situazione che potrebbe sembrare anomala, dovuta alla creazione di due connessioni separate al database, quando, invece, sarebbe stato possibile averne una sola con due recordset distinti. Nella realizzazione dell'applicazione si è voluto implementare il pattern MVC, quindi il Model deve essere separato dal controller, di conseguenza le connessioni e le operazioni con i database devono essere realizzate esternamente al controller. Questi deve richiamare le funzioni senza sapere se queste funzioni prevedono connessioni al database oppure no. Premettendo ciò, dato che si preferisce avere funzioni facilmente

riutilizzabili, deve essere possibile richiamare la funzione *verificaEsistenzaProteina()* in un'altra pagina dell'applicazione, senza dover necessariamente richiamare la funzione *inserimentoProteina()*. Cosa che non sarebbe possibile realizzare avendo le due funzioni all'interno di un'unica connessione. Nella progettazione e nella realizzazione dell'applicazione in esame è stata seguita questa indicazione.

I diagrammi di sequenza, in combinazione con i diagrammi delle classi sono necessari per poter progettare bene una applicazione Web. Tramite la realizzazione di tutti i diagrammi presenti nell'Appendice B diventa molto più semplice implementare l'applicazione, in quanto tutto ciò che deve essere realizzato è già noto. Resta solo da effettuare la "traduzione" nel linguaggio di programmazione prescelto, di tutto il contenuto informativo presente nei diagrammi.

Da questa ultima considerazione si evince l'importanza di effettuare una accurata progettazione dell'applicazione; se così non fosse, capiterebbe molte volte di implementare qualcosa di errato con la conseguente necessità di dover modificare i diagrammi e il codice stesso, con notevole aumento dei tempi di sviluppo.

CAPITOLO CINQUE

IMPLEMENTAZIONE DELL'APPLICAZIONE WEB

5.1 Premessa alla Realizzazione dell'Applicazione

5.2 Configurazione dell'Applicazione

5.3 Caratteristiche di Rilievo dell'Implementazione

Questo capitolo illustra la fase di implementazione dell'applicazione Web realizzata. Viene illustrato il processo di configurazione, proprio del framework Struts. Successivamente vengono presentate le caratteristiche di rilievo dell'applicazione, presentando degli screen-shot delle pagine realizzate.

5.1 Premessa alla Realizzazione dell'Applicazione

Si procede ora alla presentazione della fase di implementazione dell'applicazione Web. Essa è composta da due moduli, un modulo contenente le funzionalità a cui solo l'amministratore può accedere previa identificazione, l'altro contenente le funzionalità a cui tutti gli utenti possono accedere. Entrambi i moduli sono basati su interfaccia web.

Prima di iniziare con la realizzazione dell'applicazione viene creato il database *proteins* in PostgreSQL, mediante il file generato dal CASE PowerDesigner. Si procede al caricamento dei dati nel database e all'abilitazione del database alla connessione tramite TCP/IP, in quanto si ricorda che l'applicazione non è un semplice caso di studio, ma dovrà essere realmente disponibile sul web. Queste operazioni sono presentate nell'Appendice A.

Come già anticipato nel secondo capitolo, è stato utilizzato il framework Struts ed in particolare la piattaforma Eclipse[31], con l'ausilio del plug-in Exadel Studio[30]. Questa scelta garantisce una notevole compatibilità con tutti i browser presenti sul mercato: l'applicazione ha dimostrato di funzionare senza problemi con Internet Explorer e Netscape Navigator in ambiente Windows, con Mozilla in ambiente Linux.

In questo capitolo non si vuole, esclusivamente, presentare l'applicazione realizzata, bensì procedendo con la presentazione si vogliono mettere in evidenza le notevoli potenzialità del framework Struts, che nel corso dell'implementazione sono state individuate ed utilizzate.

Si effettua un'ulteriore premessa relativa all'implementazione del pattern MVC, in particolare dialogo tra Controller e Model. Il Controller al proprio interno presenta esclusivamente delle chiamate per ottenere dei risultati dal Model, senza essere a conoscenza di come il Model viene implementato e/o gestisce i dati. In particolare il Model è implementato tramite dei Java Beans che si collegano al database e tra questi i dati vengono trasferiti tramite dei DTO (Data Transfer Object).

5.2 Configurazione dell'Applicazione

Il framework Struts utilizza due diversi tipi di file di configurazione, tra loro correlati, entrambi in formato XML:

1. il file **web.xml**, detto anche *descrittore di deploy*, che non è specifico di Struts, ma è necessario per tutte le web application che utilizzano servlet. Nonostante ciò, sono previste in tale file delle informazioni specifiche per Struts. Esso viene utilizzato dal web container, quando viene avviato, per configurare e caricare le web application.

   Il formato di tale file si basa su un *Document Type Definition*[1]*(DTD)*, che definisce i tag che possono essere lecitamente utilizzati. Si procederà alla descrizione solo dei tag utilizzati nell'applicazione realizzata rimandando per ulteriori informazioni alle specifiche delle Servlet Java(http://java.sun.com/dtd/index.html).

[1] Document Type Definition(DTD), definisce le componenti ammesse nella costruzione di un documento XML "well formed". Serve a definire gli elementi leciti e la struttura che essi devono avere.

Si riporta il file web.xml realizzato per l'applicazione in esame. Il plug-in di Eclipse, Exadel Studio, fornisce la possibilità di operare sul file in una modalità ad albero(Fig. 5.1), tramite un'interfaccia amichevole oltre che nella classica versione testuale(Fig. 5.2).

Si procede alla descrizione delle varie voci presenti nella modalità testuale:

- le prime righe presentano informazioni sulla versione, sulla codifica e sul tipo di documento DTD utilizzato;

- all'interno dei tag <servlet></servlet> viene dapprima specificata la ActionServlet, che riceverà tutte le request in entrata all'applicazione, poi vengono specificati i file di configurazione specifici dell'applicazione realizzata. Si noti che i file di configurazione struts-config sono tre, ciò è dovuto al fatto che, come si vedrà in seguito, l'applicazione è stata suddivisa in tre moduli, e non in due, per motivi di leggibilità dei diagrammi. E' inoltre presente il tag <load-on-stratup></load-on-stratup>, in base al quale il container istanzia la ActionServlet allo start-up dell'applicazione;

- mediante il tag <servlet-mapping></servlet-mapping> si specifica che tutte le richieste con path terminante in .do vengono mappate sulla ActionServlet, ovvero quando c'è un URL terminante in .do viene riconosciuto dal container e la richiesta viene inviata alla Servlet di controllo;

- nel tag <welcome-file-list></welcome-file-list> viene indicata la pagina iniziale dell'applicazione;

- all'interno dei <taglib></taglib> vengono specificate le librerie di tag JSP che verranno utilizzate nel progetto;

- infine nel tag <security-constraint></security-constraint> viene specificato che qualunque richiesta di tipo Get o Post su file all'interno delle cartelle */Administrator* e */mutazioni/Administrator*, corrispondenti ai file jsp del modulo di amministrazione, non deve essere soddisfatta. In tal modo non sarà possibile accedere direttamente alle pagine presenti in tali cartelle, ma solo navigando i link presenti nella pagine precedenti.

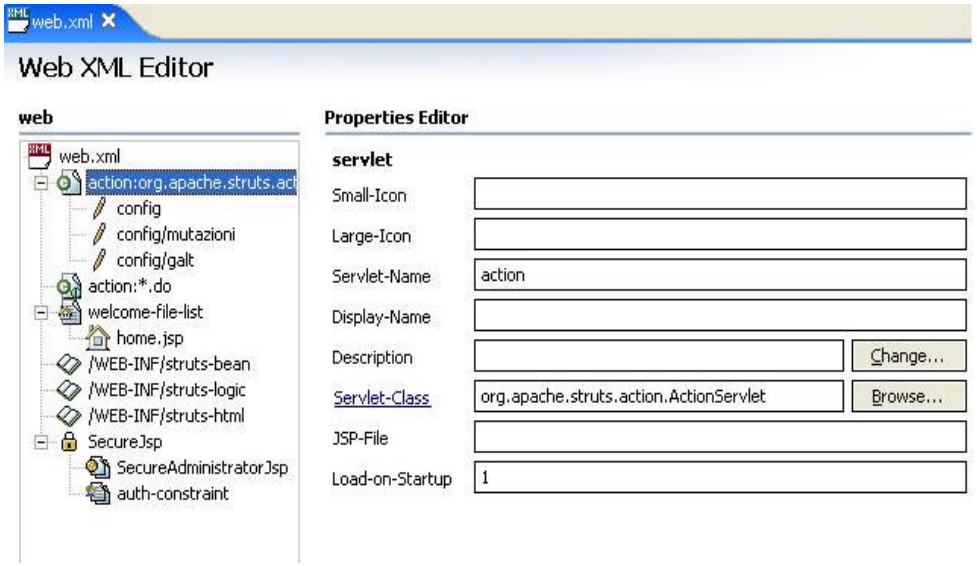


Figura 5.1 - File di configurazione *web.xml* in modalità ad albero

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
                "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
 <servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
   <param-name>config</param-name>
   <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <init-param>
   <param-name>config/mutazioni</param-name>
   <param-value>/WEB-INF/struts-config-mutazioni.xml</param-value>
  </init-param>
  <init-param>
   <param-name>config/galt</param-name>
   <param-value>/WEB-INF/struts-config-galt.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
 </servlet-mapping>
 <welcome-file-list>
  <welcome-file>home.jsp</welcome-file>
 </welcome-file-list>
 <taglib>
  <taglib-uri>/WEB-INF/struts-bean</taglib-uri>
  <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
 </taglib>
 <taglib>
  <taglib-uri>/WEB-INF/struts-logic</taglib-uri>
  <taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
 </taglib>
 <taglib>
  <taglib-uri>/WEB-INF/struts-html</taglib-uri>
  <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
 </taglib>
 <security-constraint>
  <display-name>SecureJsp</display-name>
  <web-resource-collection>
   <web-resource-name>SecureAdministratorJsp</web-resource-name>
 <description>Protect the administrator jsp</description>
   <url-pattern>/Administrator/*</url-pattern>
   <url-pattern>/mutazioni/Administrator/*</url-pattern>
   <http-method>GET</http-method>
   <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint/>
 </security-constraint>
</web-app>
```

Figura 5.2 - File di configurazione *web.xml* in modalità testuale

2. il file **struts-config.xml**, che permette di specificare in maniera dichiarativa il comportamento dei componenti del framework, evitando che tale comportamento debba essere incluso nel codice dell'applicazione. L'utilizzo di Exadel Studio ci fornisce la possibilità di creare e modificare il file in tre modalità distinte: modalità grafica (Fig. 5.3), modalità ad albero (Fig. 5.4) e modalità testuale(Fig. 5.5) classica di Struts.

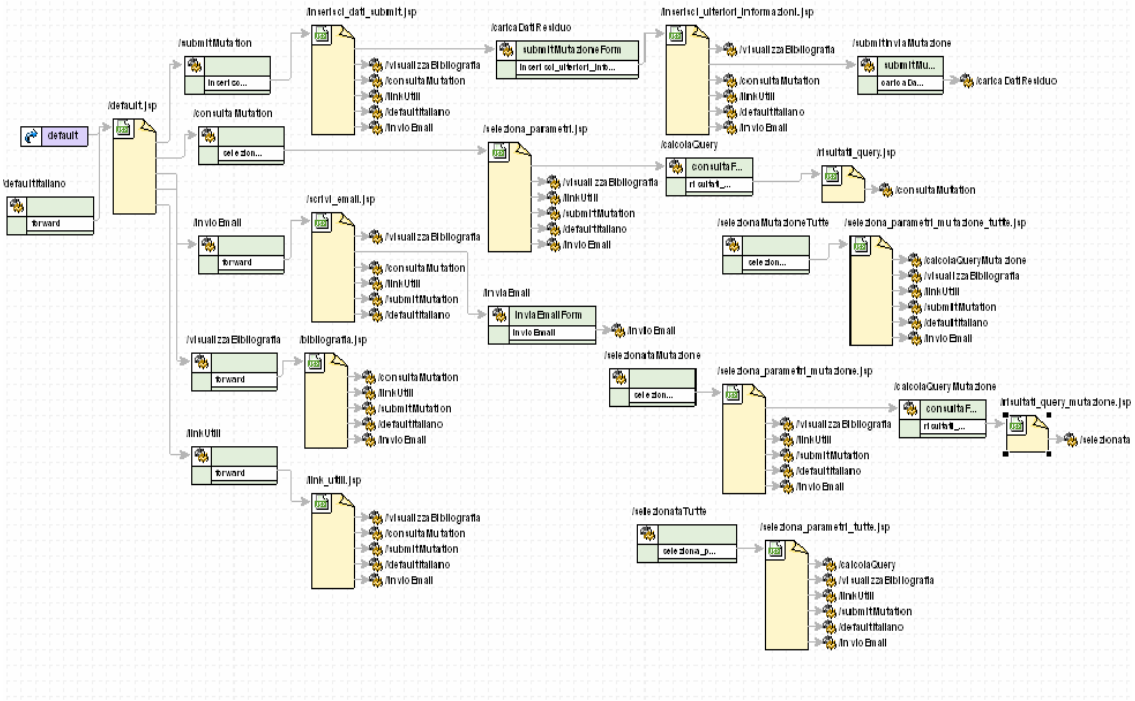


Figura 5.3 - Estratto dl file struts-config-galt.xml in modalità grafica

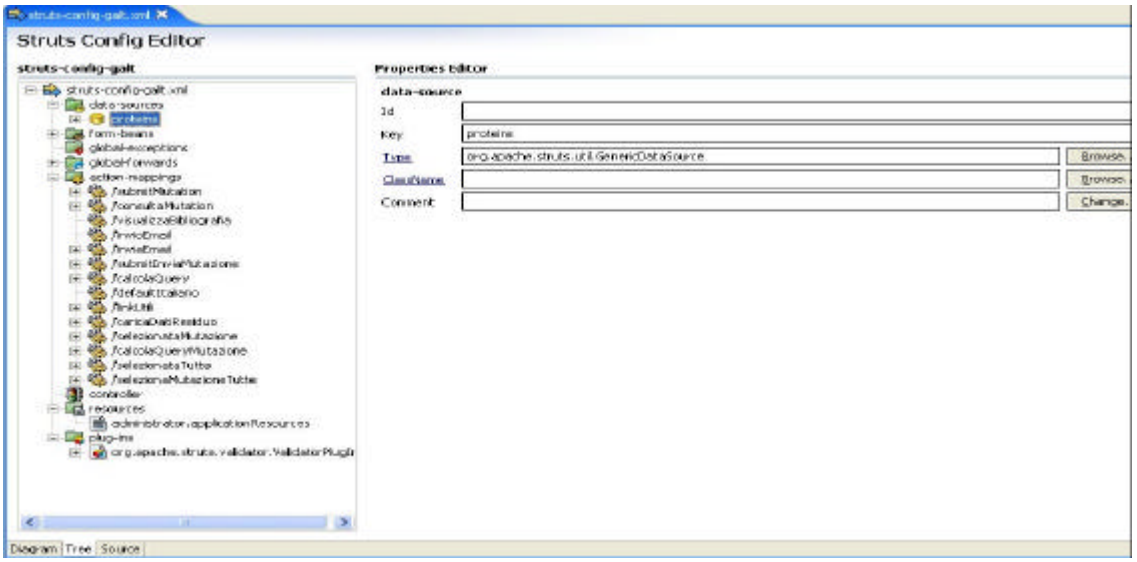


Figura 5.4 - Estratto dl file struts-config-galt.xml in modalità ad albero

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
<data-sources>
<data-source key="proteins" type="org.apache.struts.util.GenericDataSource">
<set-property property="autoCommit" value="true"/>
<set-property property="driverClass" value="sun.jdbc.odbc.JdbcOdbcDriver"/>
<set-property property="maxCount" value="10"/>
<set-property property="minCount" value="2"/>
<set-property property="url" value="jdbc:odbc:proteins"/>
</data-source>
</data-sources>
<form-beans>
<form-bean name="functionPublicBean" type="galt.functionPublicBean"/>
<form-bean name="submitMutazioneForm" type="galt.submitMutazioneForm"/>
<form-bean name="inviaEmailForm" type="galt.inviaEmailForm"/>
</form-beans>
<global-forwards>
<forward name="default" path="/default.jsp"/>
</global-forwards>
<action-mappings>
<action path="/submitMutation" scope="request"
type="galt.submitMutation" validate="no">
<forward name="inserisci_dati_submit" path="/inserisci_dati_submit.jsp"/>
</action>
<action path="/consultaMutation" scope="request"
type="galt.consultaMutationAction" validate="yes">
<forward name="seleziona_parametri" path="/seleziona_parametri.jsp"/>
</action>
<action forward="/bibliografia.jsp" path="/visualizzaBibliografia"
scope="request" type="org.apache.struts.actions.ForwardAction"/>
<action forward="/scrivi_email.jsp" path="/invioEmail" type="org.apache.struts.actions.ForwardAction"/>
<action name="inviaEmailForm" path="/inviaEmail" scope="request"
type="galt.inviaEmailAction" validate="yes">
<forward name="invioEmail" path="/invioEmail.do"/>
</action>
<action name="submitMutazioneForm" path="/submitInviaMutazione"
scope="request" type="galt.submitInviaMutazioneAction" validate="yes">
<forward name="caricaDatiResiduo" path="/caricaDatiResiduo.do"/>
</action>
<action name="consultaForm" path="/calcolaQuery" scope="request"
type="galt.calcolaQueryAction" validate="yes">
<forward name="risultati_query" path="/risultati_query.jsp"/>
</action>
<action forward="/default.jsp" path="/defaultItaliano" type="org.apache.struts.actions.ForwardAction"/>
<action forward="/link_utili.jsp" path="/linkUtili" scope="request"
type="org.apache.struts.actions.ForwardAction">
<forward name="link_utili" path="/link_utili.jsp"/>
</action>
</action-mappings>
<controller/>
<message-resources parameter="administrator.applicationResources"/>
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
<set-property property="pathnames" value="/WEB-INF/validator-rules.xml, /WEB-INF/validation-galt.xml"/>
</plug-in>
</struts-config>
```

Figura 5.5 - Estratto del file struts-config-galt.xml in modalità testuale

La modalità grafica risulta molto utile per capire i collegamenti tra i vari oggetti, ma risulta impossibile definire sorgenti dati, plug-in, form, per cui diventa necessario utilizzare almeno delle tre modalità di visualizzazione.

Anche per questo file esiste un DTD di riferimento, ma anche in questo caso non verranno specificati tutti i tag possibili, per tutti gli altri possibili parametri si rimanda al sito della fondazione Apache (http://jakarta.apache.org/commons/dtds/struts-config_1_1.dtd).
Si effettua un'analisi dei vari elementi:

- è possibile individuare in modalità grafica le pagine jsp, le action e i link tra di essi;

- nella modalità testuale e in quella ad albero è possibile individuare la sorgente dati, ovvero, nel nostro caso, la definizione del database a cui si accederà, fornendo i parametri utili alla connessione. Questo elemento si è rivelato molto utile quando si è deciso di realizzare una versione prototipo dell'applicazione con DBMS Microsoft Access, in quanto solo in questo file vengono specificati i parametri per la connessione; quando viene effettuata una query, o una qualunque operazione con il database, non bisognerà specificare nessun parametro, ma solo le seguenti linee di codice:

```
DataSource datasource;
getDataSource(request,"proteins");
Connection con = datasource.getConnection();
Statement stmt = con.createStatement();
```

- all'interno dei tag <form-bean></form-bean> sono presenti le definizioni degli ActionForm, specificando per ognuno il nome e il tipo, dove il tipo corrisponde alla classe Java che estende la ActionForm;

- nei tag <action-mappings></action-mappings> per ogni action viene descritto il mapping tra il path di una specifica request e la corrispondente classe Java che estende la classe Action; viene definito il path, il tipo, l'eventuale form associato, l'attributo validate per indicare se bisogna richiamare il metodo *validate()* del form associato, i forward, che rappresentano i path di una pagina jsp o di una Action verso cui verrà effettuato il forward, ecc. Per quanto riguarda i forward, in tale file viene effettuata una associazione tra il nome reale della pagina o della Action e un nome simbolico che verrà utilizzato in tutta l'applicazione. Se in un secondo momento si volesse cambiare un link, bisognerebbe esclusivamente cambiare il nome della pagina in tale file lasciando inalterato il nome simbolico;

- deve ovviamente essere definito il resource bundle, ovvero il file *.properties*, di default, in cui sono presenti i messaggi e i testi presenti all'interno dell'applicazione, per evitare di inserirli all'interno delle pagine jsp;

- nella parte finale del file si trovano i tag controller e plug-in. Il tag controller viene definito quando viene creato un proprio RequestProcessor, o comunque quando ne viene utilizzato uno diverso da quello di default. Il tag plug-in viene utilizzato per aggiungere

funzionalità, le quali vengono caricate all'avvio dell'applicazione e distrutte al termine. Ciò per evitare di aggiungere tali funzionalità in maniera permanente nel codice. Esempi sono il plug-in Validator o il Secure, di cui si parlerà più avanti.

Vengono ora riportati i diagrammi dei file:

- ***struts-config.xml*** (Fig. 5.6), relativo al modulo Amministrazione, ad eccezione delle operazioni sulle mutazioni;

- ***struts-config-mutazioni.xml*** (Fig. 5.7), relativo a tutte le operazioni che l'Amministratore può effettuare sulle mutazioni;

- ***struts-config-galt.xml*** (Fig. 5.8), relativo al modulo Pubblico ed in particolare alla proteina GALT.

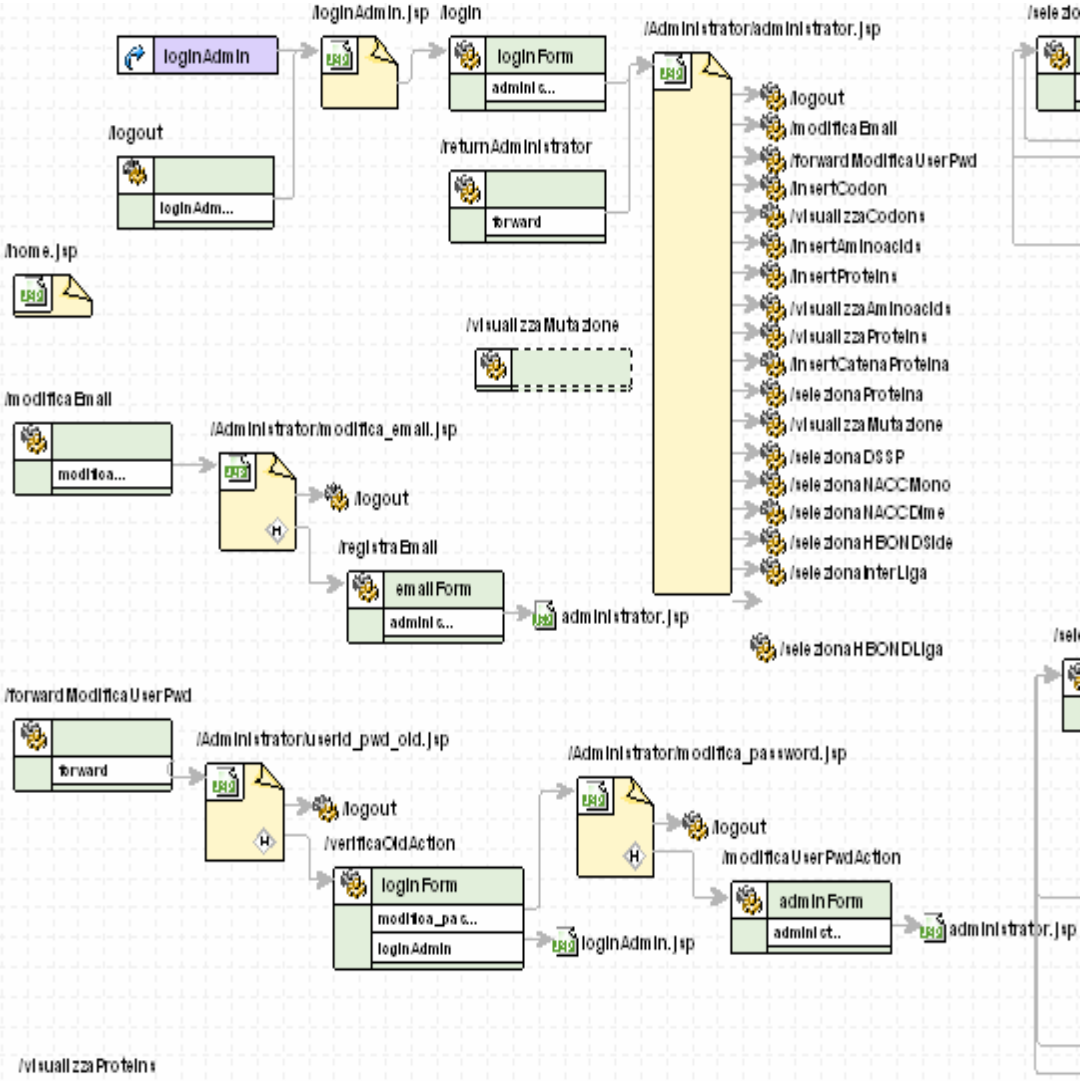


Figura 5.6 - File struts-config.xml in modalità grafica

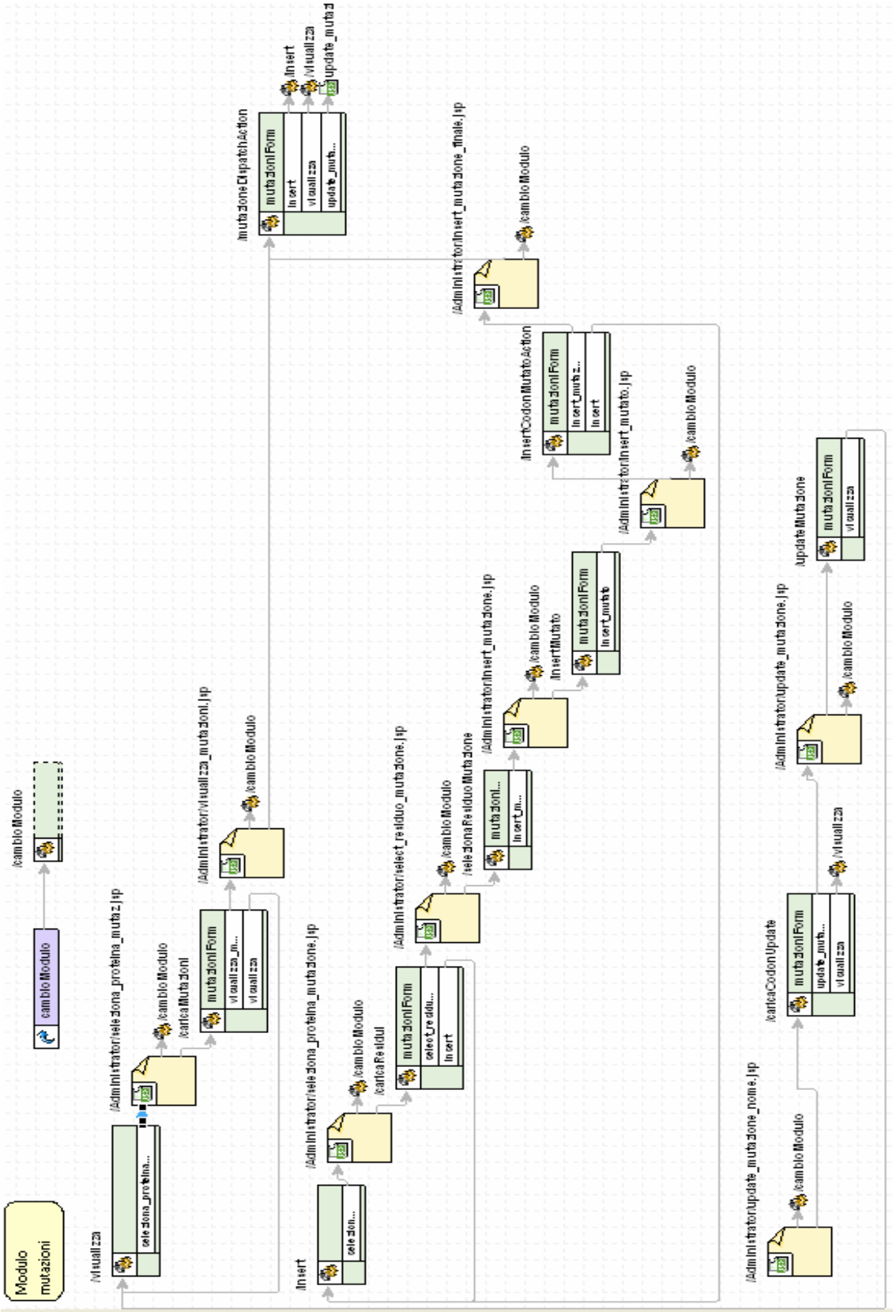


Figura 5.7 - File struts-config-mutazioni.xml in modalità grafica

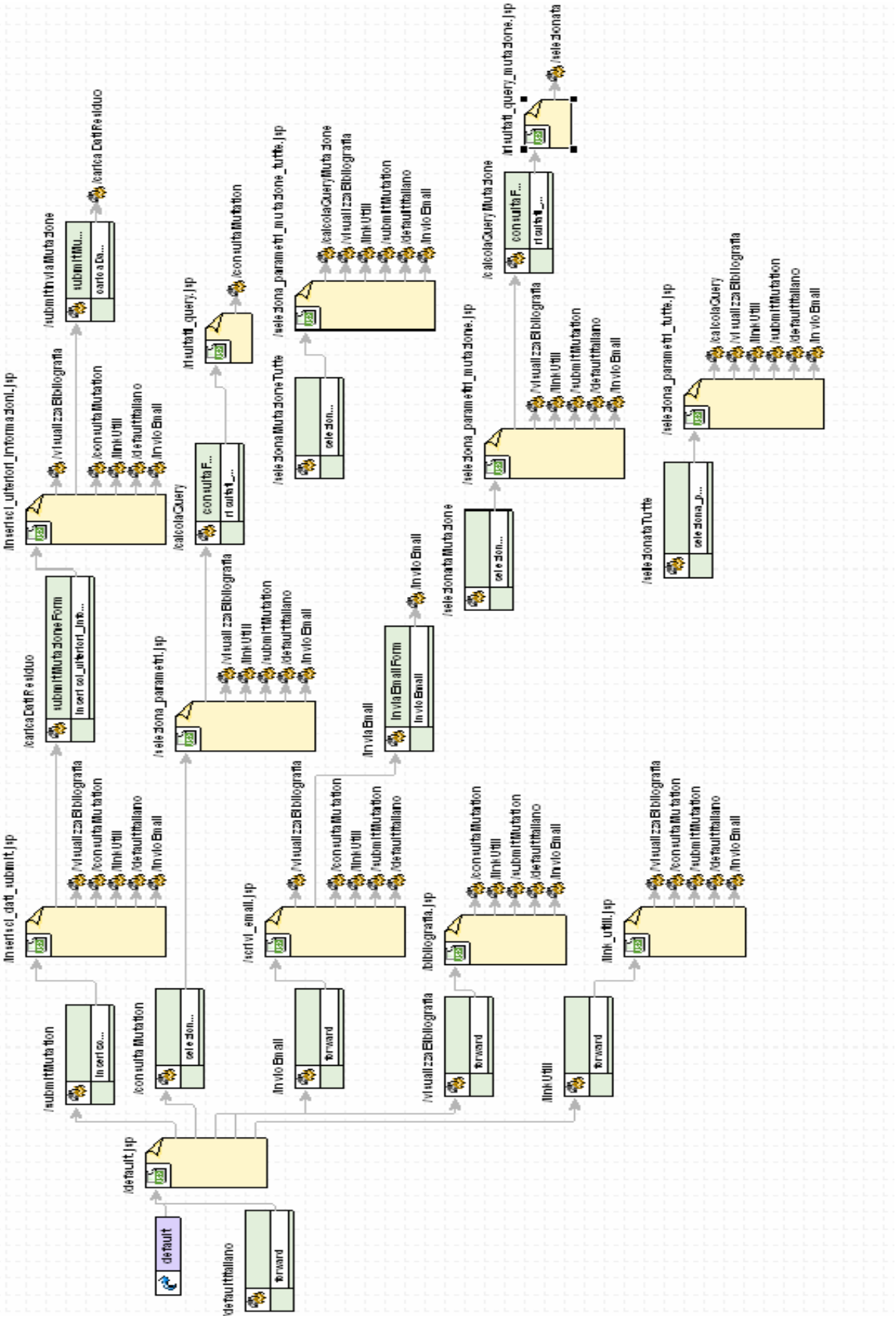


Figura 5.8 - File struts-config-galt.xml in modalità grafica

Terminata l'analisi dei file di configurazione si è passati alla scrittura di tutti i file dell'applicazione, le Action, i Form, i Bean, i DTO (*Data Transfer Object*) utilizzati per il trasferimento dei dati prelevati dal database fra i vari file, secondo i principi stabiliti nella progettazione effettuata precedentemente. Non viene riportato il codice scritto, ma viene reso disponibile, per chiunque ne fosse interessato, come materiale allegato.

5.3 Caratteristiche di Rilievo dell'Implementazione

Durante la fase di realizzazione dell'applicazione, per raggiungere tutti gli obiettivi prefissati e descritti nel primo capitolo, si sono dovuti affrontare degli aspetti di notevole importanza per la realizzazione di molte applicazioni Web. Si ritiene importante mettere in risalto alcuni di tali aspetti, separandone la trattazione.

5.3.1 Programmazione Modulare

La quasi totalità dei prodotti software è, al giorno d'oggi, realizzato da team di sviluppo. La metodologia utilizzata consiste nella suddivisione del progetto in vari moduli e nell'assegnazione di questi ai vari team, che procedono all'implementazione contemporanea in parallelo. In questo modo si prova, e in molti casi si riesce, a superare i problemi dovuti all'elevata dimensione e alla complessità dei prodotti, ma anche a ridurre i tempi di sviluppo e consegna, che fino a qualche anno fa non erano, praticamente, mai rispettati.

Con questa modalità, ovviamente, sorgono ulteriori problematiche, infatti nel ciclo di sviluppo del software deve essere introdotta, dopo la fase di

implementazione, una fase di integrazione dei vari moduli. Questa fase può essere molto laboriosa e complessa perché c'è bisogno di molta attenzione alle notazioni utilizzate dai vari programmatori, ai collegamenti tra i vari moduli, alle risorse condivise, ecc.

Nella realizzazione del framework Struts si è fatta molta attenzione nello stabilire delle direttive da seguire per la realizzazione di applicazioni suddivise in moduli, in quanto è praticamente impossibile condividere il file di configurazione *struts-config.xml*, perché esso deve essere continuamente modificato da ciascun programmatore nel corso dell'implementazione. Per tale motivo è stato realizzato un metodo molto efficace per la realizzazione di moduli separati e la successiva integrazione.

L'applicazione realizzata è stata suddivisa, a priori, in due moduli, il modulo Amministrazione e il modulo Pubblico. Ciò è tornato molto utile per illustrare le modalità di programmazione modulare. In realtà i moduli sono tre, come visto prima, ma la separazione del modulo mutazioni, da quello di amministrazione, è stata realizzata solo in fase di implementazione per motivi di gestione e visualizzazione del file di configurazione *struts-config.xml*.

Nell'applicazione il modulo di default è quello di amministrazione, il cui file di configurazione è lo *struts-config.xml*, è stato poi creato separatamente il modulo pubblico, il cui file di configurazione è lo *struts-config-galt.xml*. La realizzazione poteva essere fatta in contemporanea da due persone differenti assegnando nomi diversi per i file di configurazione e fornendo il database per effettuare le operazioni di test.

Quando i due moduli sono stati realizzati si è dovuta effettuare l'integrazione tra di essi, i passi seguiti sono stati i seguenti:

- si parte dal progetto di default, con il proprio file di configurazione e i propri file creati;

- in tale progetto si provvede alla creazione di un nuovo file Struts-Config 1.X (Fig. 5.9);

- impostando il nome del modulo del nuovo file(galt), automaticamente viene creato il tag nel file web.xml e tutti i file necessari per configurare e gestire il nuovo modulo;

- non resta che copiare, con semplici operazioni di copia e incolla le cartelle con i file creati del modulo galt, e i due moduli risulteranno perfettamente integrati.

Figura 5.9 - Aggiunta nuovo modulo al progetto

L'applicazione realizzata presenta la seguente struttura(Fig. 5.10), è possibile individuare i tre moduli, ciascuno con il proprio file di configurazione e i propri path:

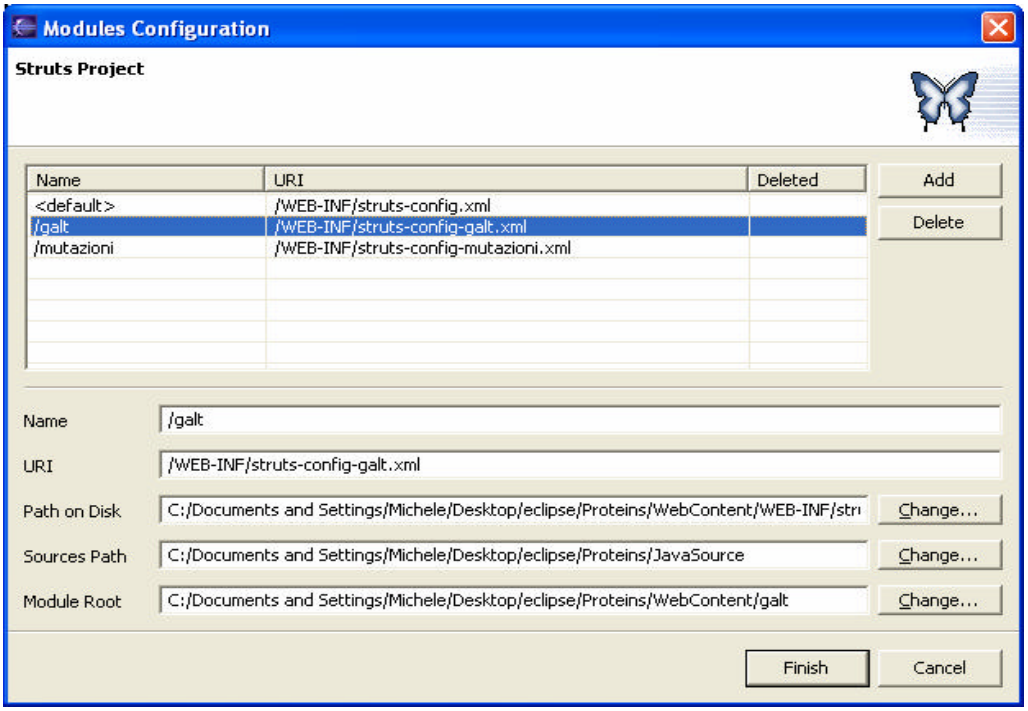


Figura 5.10 - Finestra di gestione dei moduli dell'applicazione realizzata

Resta da illustrare un'ultima operazione. Ovviamente, deve essere fornita la possibilità, ad un modulo, di richiamare delle Action o delle pagine JSP di un altro. Per fare ciò Struts mette a disposizione una particolare classe Action, la o*rg.apache.struts.actions.SwitchAction.* Nei moduli in cui è richiesta la presenza di link verso altri oggetti di altri moduli deve essere creata una Action che estende la SwitchAction, senza nessuna riga di codice all'interno.

La forma per richiamare una pagina o una Action di un altro modulo è la seguente:

/<<action>>.do?prefix=<<prefix>>&&page=<<page>>

Dove:

- <<action>> è il nome della Action che estende la SwitchAction;

- <<prefix>> è il prefisso del modulo a cui si deve passare; questo parametro è preceduto da "/" oppure può essere uguale ad una stringa vuota se si vuole passare al modulo di default;

- <<page>> rappresenta la Action o la pagina a cui reindirizzarsi una volta che è avvenuto il passaggio nel modulo indicato.

Si presenta un esempio utilizzato nell'applicazione realizzata. Nel modulo Amministrazione è presente la classe *visualizzaMutazione* che estende la SwitchAction, nel modulo mutazioni è presente una Action *insert*, che rappresenta il primo passo per effettuare l'inserimento di una nuova mutazione.

Il link, presente nella pagina *administrator.jsp* del modulo Amministrazione, è il seguente:

/visualizzaMutazione.do?prefix=/mutazioni&page=/insert.do

Il passaggio tra i vari moduli è molto efficace e non comporta aumenti di carico per il sistema.

5.3.2 Internazionalizzazione (I18N)

Facciamo una piccola premessa. Internazionalizzare un'applicazione significa predisporla a supportare lingue diverse e regioni geografiche diverse. Le specifiche standard riguardanti l'internazionalizzazione enunciano che un'applicazione internazionalizzata deve rispondere ai seguenti requisiti:

- Lingue diverse devono essere supportate senza modifiche al codice;
- Testi e immagini devono essere memorizzati esternamente al codice;
- Date, numeri, valute devono essere correttamente formattate in base alle regole della regione geografica nella quale l'applicazione è eseguita;
- Sono supportati caratteri non standard;
- L'applicazione si adatta rapidamente a supportare una nuova lingua o una nuova regione;

Il processo di adattare un'applicazione internazionalizzata ad una specifica lingua e regione si definisce *localizzazione*.

L'applicazione oggetto della tesi presenta come uno degli obiettivi principali quello di essere fruibile dal maggior numero di utenti possibili. Per tale motivo è stata scelta la lingua inglese come lingua di default per i testi ed i messaggi. Allo stesso tempo, però, si vuole fare in modo che essa sia visionabile anche in italiano e, in futuro, in altre lingue straniere.

In base ai metodi di programmazione standard si deve essere a conoscenza delle lingue in cui realizzare l'applicazione già nella fase di progettazione, perché l'aggiunta di una nuova lingua comporta delle modifiche all'architettura del progetto e anche al codice. In tali condizioni diventa molto difficile l'introduzione futura di nuove lingue.

Le classi Java fondamentali per la localizzazione di una applicazione predisposta all'internazionalizzazione sono due: java.util.Locale e java.util.ResourceBundle.

- La classe java.util.Locale è quella che consente all'applicazione di conoscere in quale lingua e in quale paese viene eseguita, in pratica identifica il particolare linguaggio in base ai codici di lingua e regione impostati nel sistema. Ad esempio i codici che identificano la lingua italiana e il paese Italia sono "it" e "IT";

- La classe java.util.ResourceBundle è invece un contenitore di oggetti cosiddetti locale-specific, ovvero oggetti che assumono valori differenti in base al locale. Un esempio di risorsa locale-specific può essere una stringa; l'applicazione internazionalizzata verifica il valore del Locale e in base a

questo reperisce la stringa dal ResourceBundle specifico. In realtà la classe java.util.ResourceBundle è una classe astratta. Una implementazione concreta è la java.util.PropertyResourceBundle che gestisce le risorse specifiche per un dato locale memorizzandole in file di properties.

Fra tutti i requisiti, definiti prima, che un'applicazione internazionalizzata deve soddisfare, Struts fornisce il supporto essenzialmente per ciò che riguarda il reperimento di testo e immagini localizzate, che risulta molto semplice ed efficace. Per tutti gli altri aspetti, quali formattazione di date, importi etc. bisogna fare ricorso a classi Java standard per le quali si rimanda alla documentazione ufficiale presente sul sito della Sun(http://java.sun.com).

Struts gestisce l'internazionalizzazione fornendo gli strumenti per reperire le risorse localizzate nei resource bundle, i file properties, in base al locale corrente.

In una web-application realizzata con Java, e quindi anche in quelle costruite con Struts, è possibile reperire l'informazione relativa al locale dell'utente mediante il metodo public della classe java.util.Locale *getLocale()* all'interno di una servlet, oppure tramite il custom tag <*html:html locale="true"*> all'interno di una pagine JSP realizzata con Struts; una volta reperita l'informazione essa viene memorizzata per la durata di una sessione.

Per la gestione dei resource bundles in Struts viene usata la classe org.apache.struts.util.MessageResources che segue la stessa logica della java.util.ResourceBundle, quindi anche la classe org.apache.struts.util.MessageResources è una classe astratta, e la sua concreta implementazione è fornita dalla classe org.apache.struts.util.PropertyMessageResources che consente di leggere stringhe localizzate alle quali è associata una chiave reperendole dai file di properties.

Gli elementi di base per l'internazionalizzazione di un'applicazione Struts sono quindi i file .properties.

Nell'applicazione in esame sono presenti due di questi file (Fig. 5.11):

- *ApplicationResources.properties*, che contiene i testi e i messaggi nella lingua di default(inglese);

- *ApplicationResources_it.properties*, che contiene i testi e i messaggi nella lingua italiana.

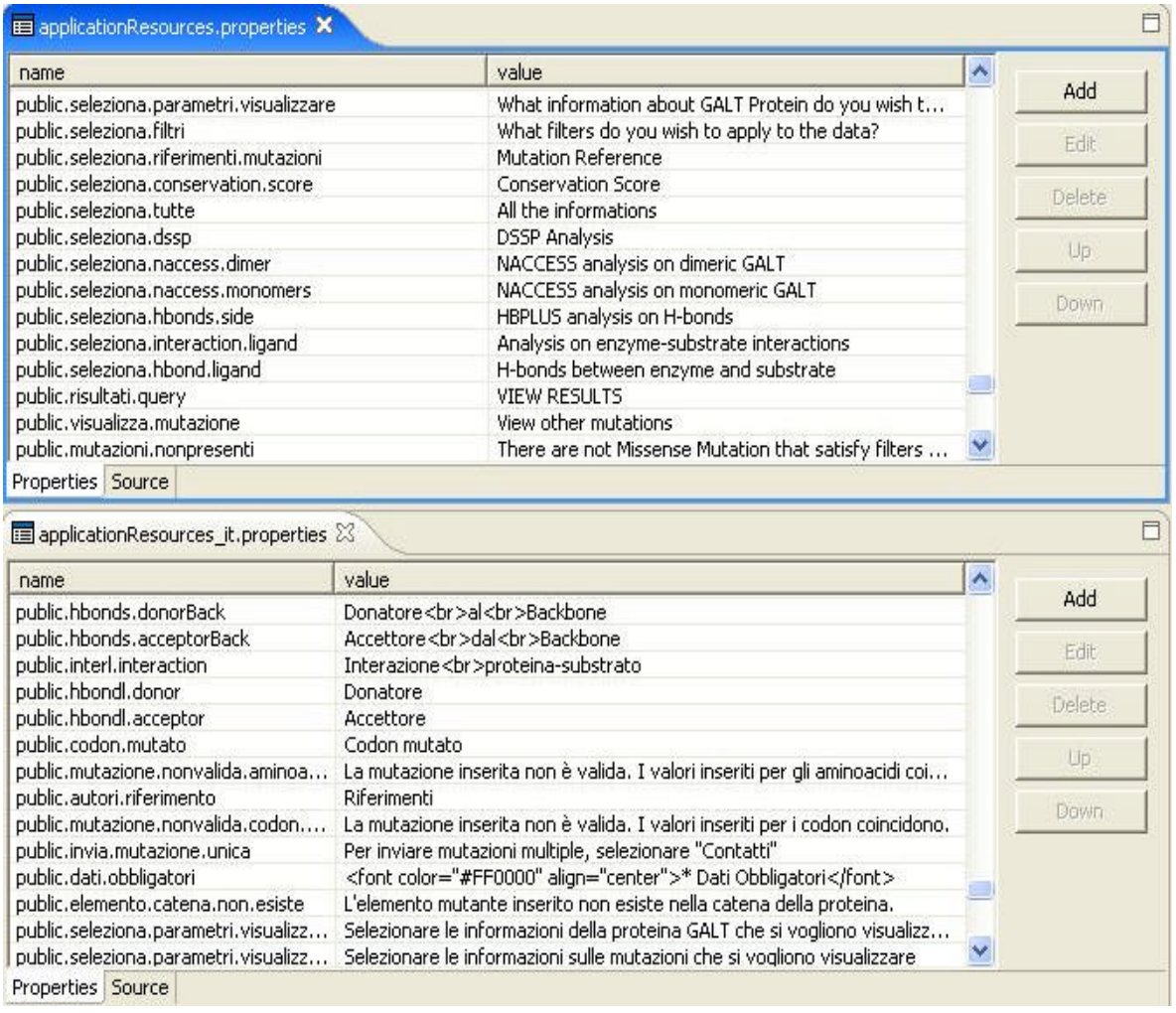


Figura 5.11 – Resource bundles dell'applicazione

Come si può notare dall'immagine l'attributo name dei due file contiene lo stesso valore, mentre per quanto riguarda l'attributo value, esso è stato inserito in modo diverso in base alla lingua. Per visualizzare uno di questi messaggi in una pagina jsp si utilizza la classe di tag *bean*, in particolare:

<bean:message key="public.dati.obbligatori"/>

Dove key corrisponde al valore dell'attributo name.

Si può ora procedere alla descrizione del principio di funzionamento dell'internazionalizzazione in Struts.

Tutti i resource bundle devono essere localizzati nella stessa cartella. All'interno del file di configurazione *struts-config.xml* deve essere inserito solo il resource bundle di default, come attributo *parameter* del tag <*message-resources*>.

Quando si carica l'applicazione, viene caricato il file struts-config.xml, e quindi individuata la posizione del resource bundle; nel momento in cui viene richiamata una Action o una pagina jsp viene prelevato il locale al suo interno, tramite i metodi visti prima, e viene salvato nella sessione; se si incontra il tag per la visualizzazione di un messaggio o di un testo viene cercato il resource bundle relativo al locale presente in sessione, se esso non viene trovato si utilizza quello di default.

Per testare l'applicazione Web realizzata è stata caricata la home-page impostando come lingua di sistema il tedesco(Fig. 5.12).

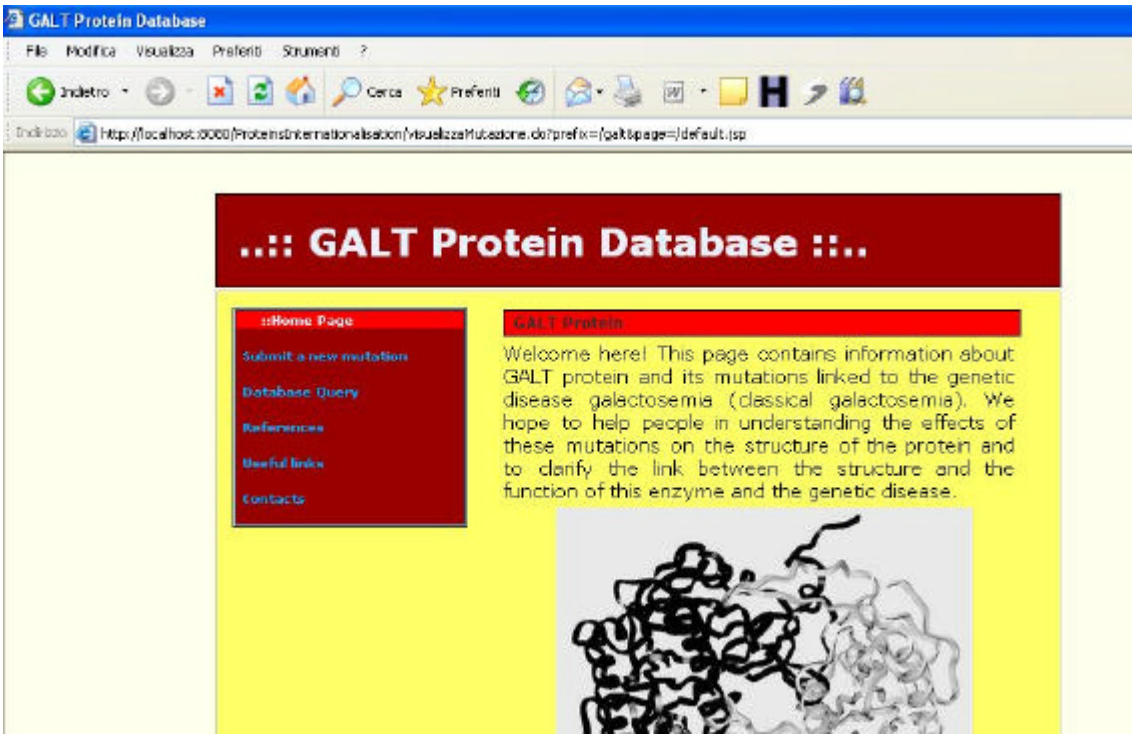


Figura 5.12 - Home page con lingua impostata: tedesco

Come si può notare tale pagina è in inglese, ciò perché non è presente un resource bundle in tedesco, ora si provvede ad impostare la lingua di sistema ad Italiano(Fig. 5.13) e a ricaricare la pagina per visualizzare il risultato(Fig. 5.14):

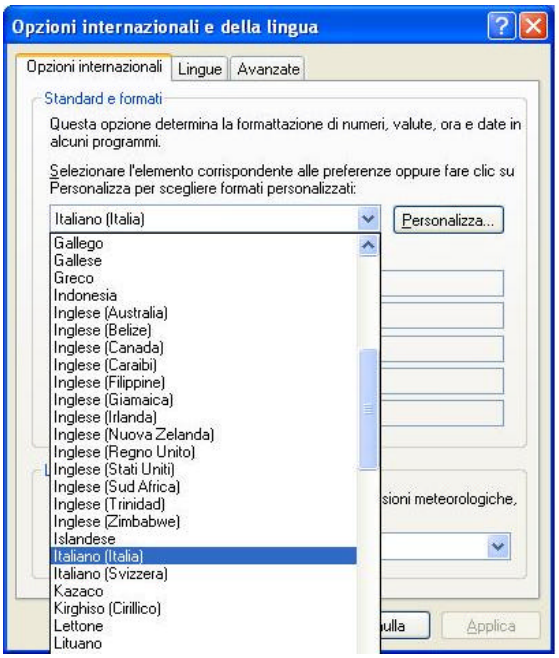


Figura 5.13 - Impostazione della lingua Italiano

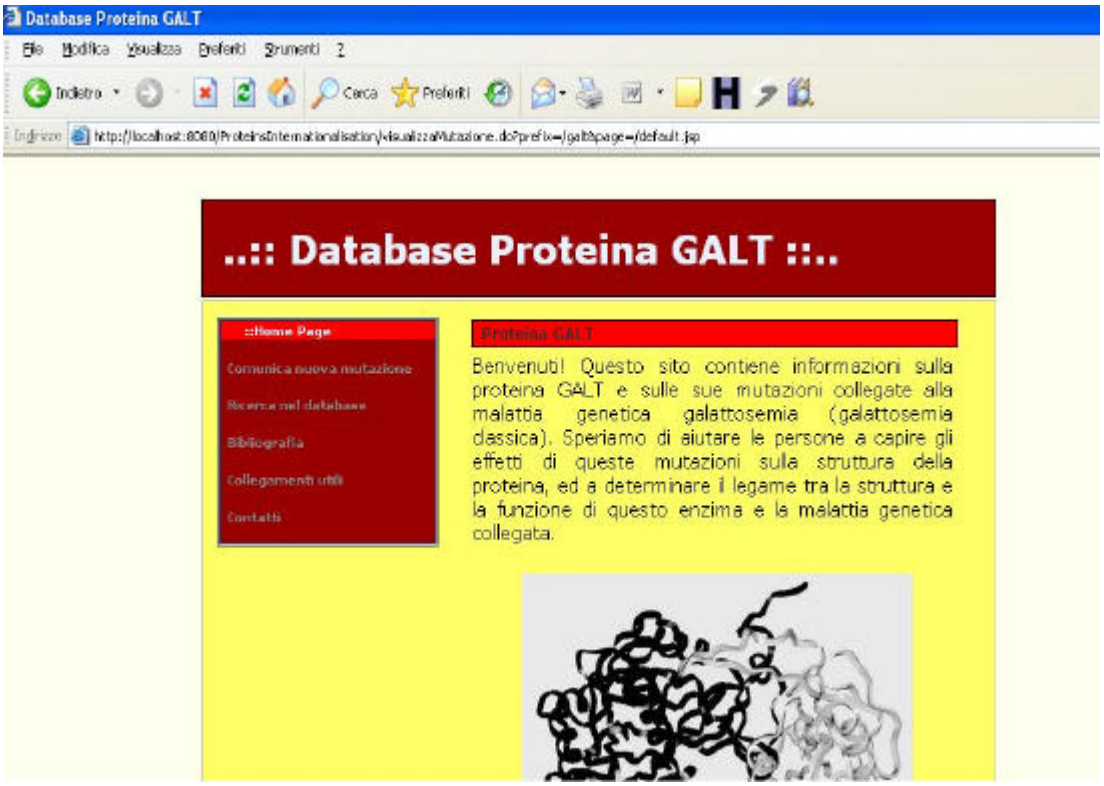


Figura 5.14 - Home page con lingua impostata: italiano

E' ovvio che la localizzazione funziona perfettamente. Per aggiungere un'ulteriore lingua, in futuro basterà creare un nuovo resource bundle, assegnandogli un nome opportuno in base al locale.

Si può quindi asserire con certezza che l'utilizzo dei resource bundle fornisce la possibilità di risolvere due problematiche che spesso si presentano ai produttori di applicazioni Web:

1. la facile Manutenzione, in quanto nel caso in cui si dovesse modificare un messaggio, anche se in un futuro lontano, presente in più pagine basterebbe semplicemente modificare il parametro value all'interno del file .properties;

2. la gestione di più lingue senza alcuna aggiunta nel codice e/o nell'architettura.

5.3.3 Validazione dei Dati Lato Client e Lato Server

Per effettuare la validazione dei dati immessi in un form HTML, il framework Struts mette a disposizione degli sviluppatori il metodo *validate()* della classe ActionForm; così facendo è possibile effettuare la validazione lato Server.

Questo meccanismo semplifica notevolmente il compito degli sviluppatori, in quanto non devono inserire codice all'interno della logica ogni volta che vengono letti i dati da un form, ma presenta alcuni inconvenienti. Innanzitutto il codice di validazione deve essere duplicato anche se deve essere fatta la stessa verifica su campi del form differenti, inoltre una modifica delle regole di validazione comportano una modifica ai sorgenti dell'applicazione. Per ovviare a questi problemi è stato sviluppato, all'interno del progetto Jakarta della fondazione

Apache, **il framework Validator**, il quale viene fornito all'interno del framework Struts.

Tale framework consente di effettuare validazione lato Client e/o lato Server in modo automatico e dichiarativo, tramite l'utilizzo di due file xml, senza dover apportare modifiche al codice. La configurazione delle routine di validazione da applicare ad un campo di un form è fatta mediante un file di configurazione XML, quindi esternamente all'applicazione ed è facilmente modificabile al mutare delle esigenze applicative.

5.3.3.1 Il Framework Validator

Il Validator [41] è costiuito da un insieme di classi predisposte per eseguire tutti i più comuni controlli di validazione in genere usati nelle applicazioni, ma esiste anche la possibilità di creare routine di validazione proprie.

Come già anticipato, il framework opera mediante due file xml:

- *validator-rules.xml*, in cui vengono dichiarate tutte le routine di validazione disponibili, i loro nomi logici e il codice JavaScript corrispondente a ciascuna routine di validazione per l'esecuzione dei controlli;

- *validation.xml*, in cui si specifica come queste routine vengano applicate ai vari campi di input dei form dell'applicazione, ai quali si fa riferimento mediante i nomi dei form beans dichiarati nello struts-config.xml.

Per ogni modulo di un'applicazione realizzata con Struts è presente una copia questi file, opportunamente rinominati in base al modulo. Pertanto ogni sviluppatore ne ha una propria versione e non deve preoccuparsi della configurazione o delle notazioni utilizzati dagli altri.

5.3.3.2 Configurazione del Validator per l'utilizzo con Struts

Utilizzare il Validator all'interno di una applicazione realizzata con Struts significa eseguire i seguenti step:

1. Abilitare il Validator plug-in;

2. Configurare i due file XML appena citati, validator-rules.xml e validation.xml;

3. Creare form bean che estendano gli ActionForm del Validator.

Analizziamo nel dettaglio le varie fasi:

1. Il Validator viene configurato come un plug-in di Struts. Per abilitare il Validator bisogna aggiungere nello struts-config.xml le seguenti righe, al fine di far caricare e inizializzare il Validator all'avvio dell'applicazione e di definire i

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

i percorsi dei file di configurazione.

2. Nel file *validator-rules.xml* sono presenti una serie di routine(Fig. 5.15), fornite dalla versione standard del Validator, che coprono la gran parte delle comuni esigenze per ciò che riguarda i controlli formali da eseguire sui campi di input di un form. Il file in genere non deve essere modificato a meno che non si vogliano definire delle proprie routine.

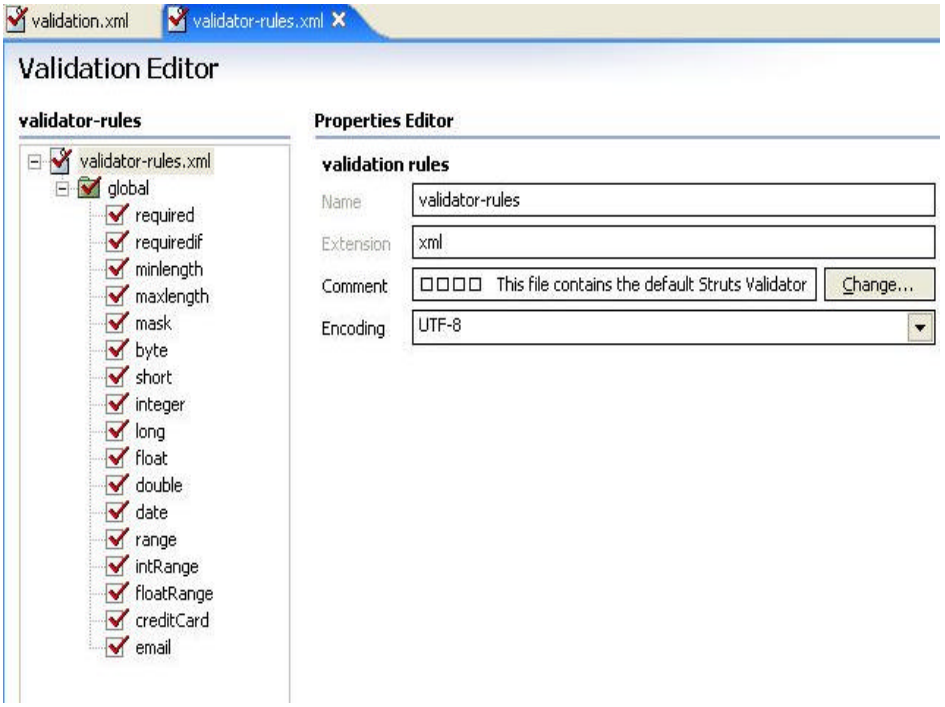


Figura 5.15 - File validator-rules.xml, nella modalità ad albero fornita da Exadel

Come si può vedere ad ogni regola è associato un nome logico, che sarà usato come riferimento nel file validation.xml che vedremo. A ciascuna di queste regole è associato il codice JavaScript necessario per effettuare la validazione.

Il file *validation.xml* viene usato per dichiarare i controlli che verranno effettuati sui campi di input di un form dell'applicazione. Si riporta il codice del file *validation.xml* dell'applicazione realizzata relativo al form utilizzato per effettuare il Login nel modulo Amministrazione:

```
<form name="loginForm">
 <field property="username" depends="required">
  <arg0 key="username" resource="false"/>
 </field>
 <field property="password" depends="required">
  <arg0 key="password" resource="false"/>
 </field>
</form>
```

Per ogni form dell'applicazione va definito un elemento <form></form> in cui l'attributo name corrisponde al nome del form bean dichiarato nello struts-config.xml. All'interno del tag <field></field> si dichiarano i controlli da eseguire per ogni campo del form. Nell'esempio si dichiara che i campi username e password sono obbligatori mediante l'attributo depends. Il parametro resource, se impostato a true, indica che è presente un messaggio nel resource bundle associato al particolare argomento.

Per i messaggi di errore associati a ciascuna routine di validazione il Validator utilizza il file di risorse .properties di cui si è ampiamente discusso prima. Non è specificato l'identificativo del messaggio perché esso è compreso all'interno del codice di validazione JavaScript. Ad esempio il messaggio per la proprietà required è associato all'identificativo: errors.required;

L'ultima annotazione da effettuare a questo proposito è relativa alla gestione modulare. Non avendo introdotto nuove regole di validazione

nell'applicazione realizzata, il file di configurazione *validation-rules.xml* è lo stesso per tutti e tre i moduli. Quello che varia è il file validation.xml, in quanto ne esistono tre versioni, una per modulo:

- *validation.xml*, relativamente al modulo con file di configurazione struts-config.xml, ;

- *validation-mutazioni.xml*, relativamente al modulo con file di configurazione struts-config-mutazioni.xml;

- *validation-galt.xml*, relativamente al modulo con file di configurazione struts-config-galt.xml;

Nella figura 5.16 si riportano i form per cui sono stati impostati dei controlli, relativamente a tutti e tre i file.

Ciascuno di questi tre file è indipendente dagli altri, quindi nella realizzazione separata di ogni modulo non ci si è dovuti preoccupare di quanto presente nei restanti due.

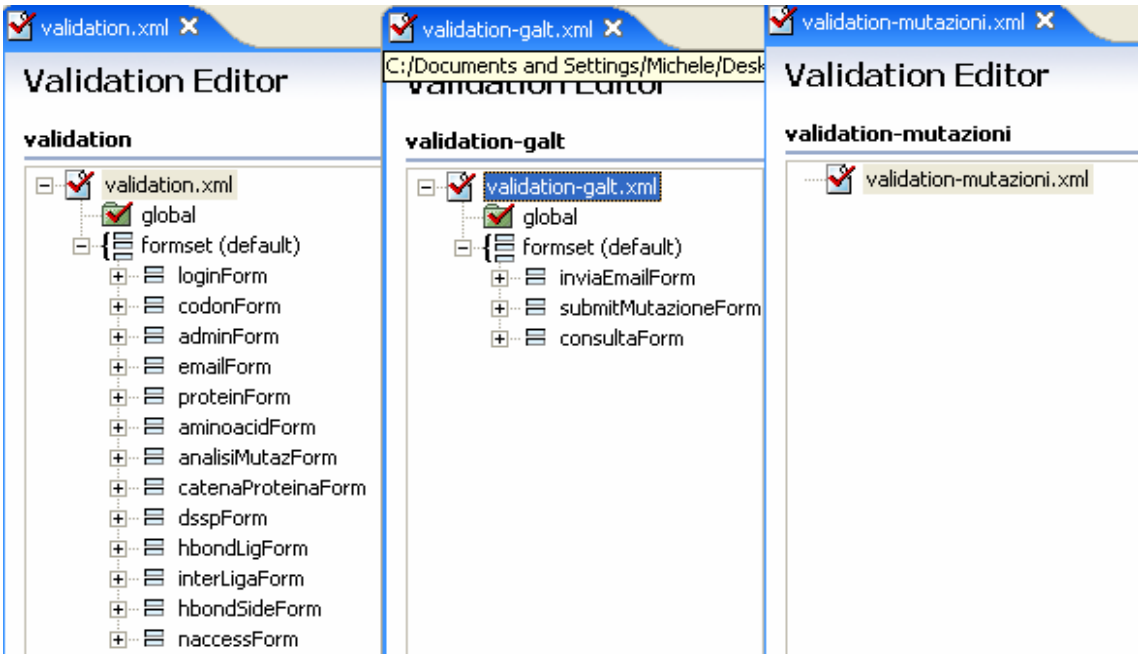


Figura 5.16 - Form validati nei tre moduli

3. Per completare la configurazione del Validator non resta che creare i form bean estendendo le calssi del validator. In tal caso i form non devono estendere la classe *org.apache.struts.action.ActionForm*, bensì la classe *org.apache.struts.validator.ValidatorActionForm* che fornisce un'implementazione di metodi *validate()* e *reset()* propri.

Completata l'abilitazione del framework validator, bisogna decidere se effettuare la validazione lato Client o lato Server. La validazione lato Server prevede che il Submit dei dati avvenga sempre, quindi prevede che i controlli avvengano solo nel server. Nel caso in cui i controlli non vadano a buon fine, la pagina contenente il form viene proposta nuovamente all'utente, con il messaggio di errore, per far re-inserire i dati. Tutto ciò comporta al server degli appesantimenti che possono essere evitati con la validazione lato Client. Validando i dati sul Client il Submit non avviene se i dati non sono inseriti correttamente, solo al verificarsi di tale situazione i dati vengono passati al

Server. Anche in questa occasione, comunque, il Validator esegue anche i controlli lato-server una volta che il submit del form viene effettuato. Ciò garantisce che i controlli formali vengano effettuati anche nel caso in cui i controlli JavaScript lato client vengano disattivati, cosa non da poco visto che in genere nelle applicazioni web che eseguono i controlli sui campi lato client , le validazioni non vengono eseguite anche lato server come in realtà dovrebbe essere fatto. Ciò conferisce una maggiore robustezza ed affidabilità all'applicazione ed è uno dei principali benefici dell'uso del Validator.

Per attivare l'abilitazione lato Client, bisogna apportare due modifiche nella pagine Jsp in cui è presente la definizione del Form:

- Introdurre il tag *<html:javascript formName="nomeForm"/>*;

- Modificare il tag di definizione del form in

<html:form action="nomeAction.do" onsubmit="return validateNomeForm (this)">

Il Validator genererà automaticamente la funzione javaScript *validateNomeForm()* e la inserirà nel codice quando verrà fatto il build dell'applicazione.

Nell'applicazione realizzata si è usata esclusivamente la validazione lato Client. Si mostra la risposta del sistema se nel form dove viene effettuato il Login vengono lasciati vuoti i due campi, per cui è stata definita la proprietà required (Fig. 5.17).

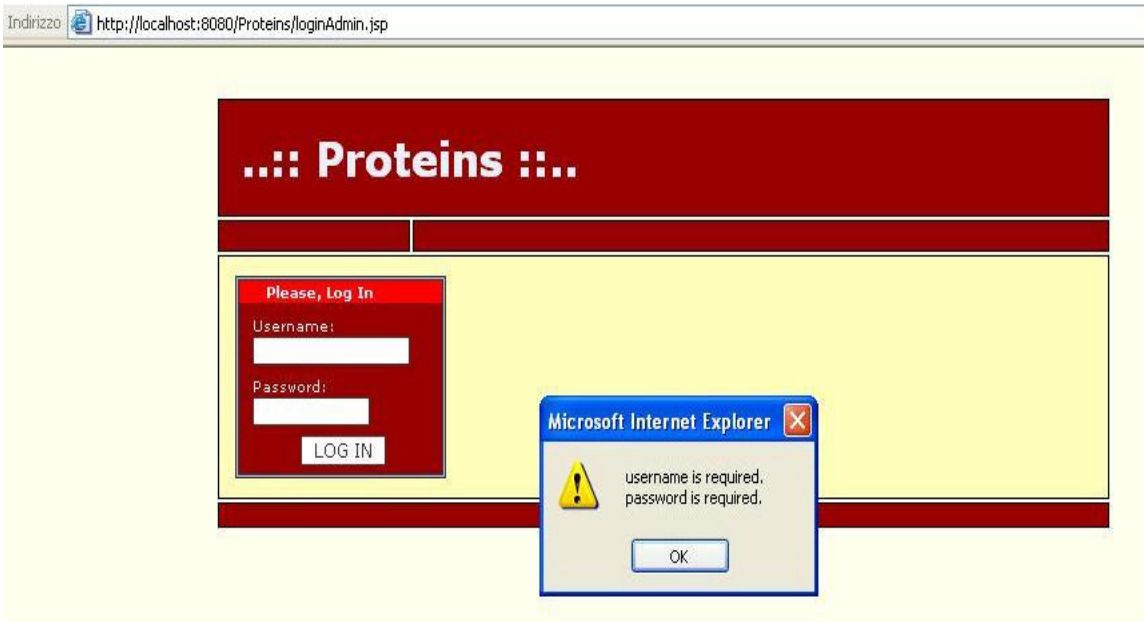


Figura 5.17. Validazione dei dati lato Client con l'utilizzo del Validator

Se non si fosse utilizzata tale soluzione, i dati sarebbero stati inviati al server, verificati e solo allora sarebbero stati inoltrati i messaggi di errore.

5.3.4 Gestione della Sicurezza

Gli sviluppatori di applicazioni Web si trovano, sempre più frequentemente, a dover affrontare il problema della "Sicurezza Informatica", ovvero il dover salvaguardare i dati sensibili del software che si realizza, da potenziali rischi e/o violazioni dei dati.

Aspetti fondamentali della sicurezza sono:

- *L'autenticazione e l'autorizzazione degli utenti*: nella maggior parte delle applicazioni esistono alcune funzionalità che devono essere accedute solo da determinati e noti utenti. L'autenticazione di un utente consiste nella

verifica dell'identità dell'utente. La fase immediatamente successiva consiste nello stabilire le risorse alle quali l'utente può accedere e con quali modalità operative;

- *La trasmissione sicura dei dati sulla rete*: esistono particolari prodotti software(gli *sniffer*) in grado di intercettare e registrare il traffico che viene trasmesso sulla rete, con l'obiettivo di ricavare ciò che è stato trasmesso, come ad esempio codici e password, tramite la successiva analisi di tutto ciò che è stato intercettato. Per tale motivo i dati sensibili delle applicazioni vengono trasmessi tramite connessioni sicure (HTTPS = HTTP Secure).

Nell'applicazione realizzata è presente il modulo di Amministrazione che deve essere accessibile al solo utente Amministratore, inoltre si preferisce che i dati trasmessi tra questi e il Server, utilizzino una connessione sicura.

Vengono di seguito presentate le modalità di gestione della sicurezza realizzate nell'applicazione oggetto della tesi, mostrando i vantaggi di cui è stato possibile usufruire adottando il framework Struts per l'implementazione.

5.3.4.1 Autenticazione

Per quanto riguarda l'autenticazione, è stata prevista una relazione nel modello del database contenente lo *username* e la *password* dell'amministratore. Per entrare nel modulo di Amministrazione bisogna quindi riempire il form in figura 5.17 con i dati corretti, inoltre bisogna effettuare la connessione da uno degli indirizzi dell'*Istituto di Scienze dell'Alimentazione del CNR di Avellino*,

ovvero con un IP che inizia per "140.164.41". Questo ulteriore controllo è stato richiesto espressamente dal committente dell'applicazione. Nel caso in cui tutte le condizioni previste siano verificate, l'Amministratore viene abilitato ad eseguire le operazioni previste e viene caricata la home page del modulo di Amministrazione(Fig. 5.18).

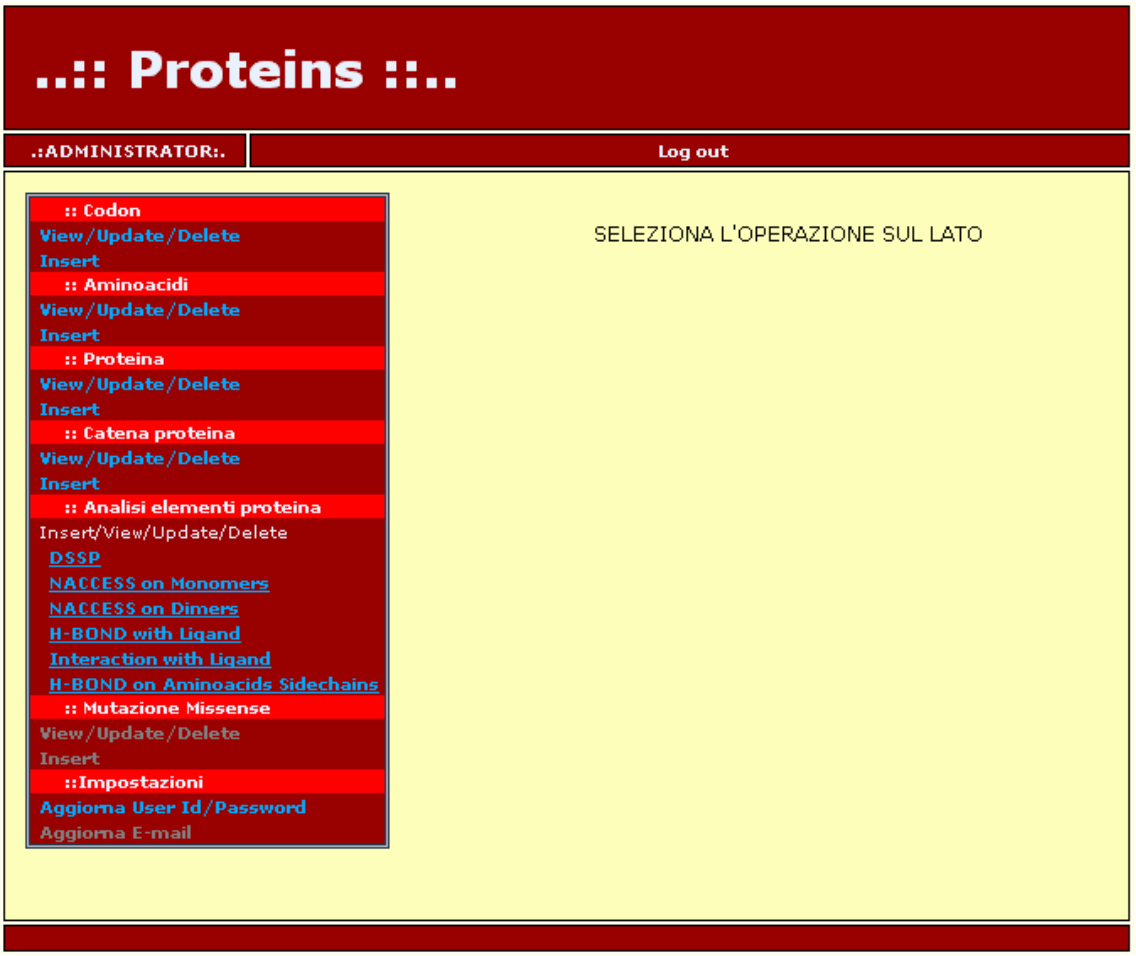


Figura 5.18 - Home page del modulo di Amministrazione

5.3.4.2 Autorizzazione

Se il processo di autenticazione va a buon fine viene memorizzato un attributo in una variabile di sessione. Questo attributo viene verificato prima dell'esecuzione di ogni Action per evitare che un utente non autorizzato riesca ad entrare nelle pagine di amministrazione introducendo il link diretto.

Per evitare di dover inserire il controllo in ogni Action, Struts fornisce la possibilità di creare una classe che estende la classe RequestProcessor, vista in precedenza, ed in particolare è uso comune fare l'override del metodo *processPreprocess()*. Tale metodo non contiene nulla di default, è stato creato appositamente per inserire eventuali controlli prima di ogni Action. Esso viene eseguito dal Request Processor prima dell'elaborazione di ogni richiesta, quindi di conseguenza è possibile introdurre il controllo delle autorizzazioni prima dell'esecuzione di qualunque elaborazione nel seguente modo.

```
public class customRequestProcessor extends org.apache.struts.action.RequestProcessor {

   protected boolean processPreprocess(HttpServletRequest request,
         HttpServletResponse response) {
      HttpSession session = request.getSession( );
      String michele = (String) session.getAttribute("amministratore");
      if (michele == null) {
         try {
            response.sendRedirect("loginAdmin.jsp");
         } catch (IOException e) {
            log.error("Unable to send response");
         }
         return false;
      }
      return true;
   }
}
```

Come si può vedere, se l'utente non è autorizzato viene effettuato un redirect alla pagina *loginAdmin.jsp*, pertanto per accedere alle pagine di amministrazione bisogna obbligatoriamente passare per la pagina in cui è presente il form per il Login e superare tutti i controlli di l'autenticazione.

Per quanto riguarda le pagine JSP non è necessario effettuare un tale tipo di controllo, in quanto all'interno del file web.xml, riportato in precedenza, è stato stabilito che non è possibile accedere, tramite link diretti, a tali pagine. Ciò implica che esse possano essere accedute solo tramite la navigazione dei link all'interno del sito.

Chi provasse ad accedere direttamente ad una di queste pagine si troverebbe di fronte una pagina di errore indicante l'impossibilità a soddisfare la richiesta (Fig. 5.19).

Figura 5.19 - Pagina visualizzata se viene inserito un link diretto ad una pagine JSP del modulo Amministratore

5.3.4.3 Trasmissione dei Dati su Connessione Sicura

Relativamente al modulo Amministrazione, si è preferito effettuare la trasmissione dei dati in "modalità sicura", ovvero senza che nessuno abbia la possibilità di decifrarli oltre al Client e al Server. Lo standard *de facto* per le trasmissioni sicure tra Client e Server è il protocollo HTTPS, tale termine indica l'utilizzo del protocollo HTTP con il supporto SSL[2]. Prima di analizzare come la trasmissione è stata realizzata nell'applicazione, si provvede ad illustrare il funzionamento di tale trasmissione.

- ***Principio di Funzionamento del Protocollo HTTPS***

  La trasmissione di dati con il protocollo HTTPS avviene tramite una procedura a due fasi:

  I. Creazione connessione SSL: viene effettuato uno scambio di messaggi e chiavi tra client e server al fine di creare un canale di comunicazione criptato, ovvero stabilire una connessione SSL. L'algoritmo utilizzato è quello della *crittografia a chiave pubblica*;

[2] Secure Socket Layer (SSL) è un protocollo progettato dalla Netscape Communications Corporation per realizzare comunicazioni cifrate su Internet. Questo protocollo utilizza la crittografia per fornire sicurezza nelle comunicazioni su Internet e consente alle applicazioni client/server di comunicare in modo tale da prevenire la manomissione dei dati, la falsificazione e l'intercettazione. All'interno del modello a Stack di Internet tale Layer è posto al di sotto del livello applicativo(protocolli http, smtp, ecc.) e al di sopra del livello di trasporto (protocollo TCP).

II. Trasmissione dei dati: si effettua la trasmissione dei dati tra il client e il server in modo sicuro sul canale creato, tramite l'utilizzo del protocollo HTTP.

E' interessante analizzare più in dettaglio lo scambio di messaggi e chiavi [44], effettuato durante la prima fase(Fig. 5.20):

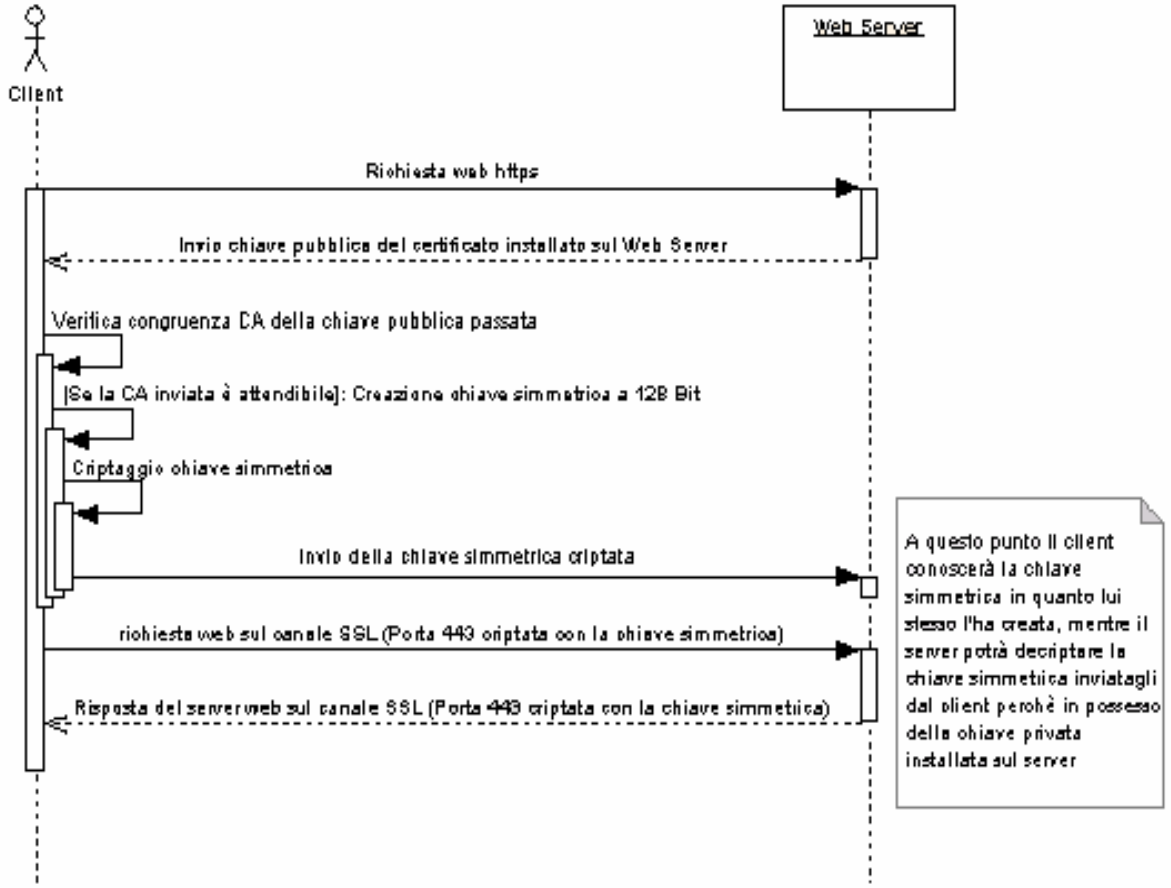


Figura 5.20 - Creazione della connessione SSL

1. Il client richiede il servizio tramite il suffisso *https://*;

2. Il web server invia al client la chiave pubblica del certificato installato;

3. Il client verifica se il certificato inviato dal web server è garantito da una specifica Authority (CA Pubblica o CA Privata) e comunica all'utente tramite interfaccia la possibilità di accettare o di non accettare la connessione;

4. Se accetta la connessione il client crea una chiave simmetrica a 128 Bit che servirà per criptare e decriptare i dati;

5. Il client cripta la chiave simmetrica con la chiave pubblica inviata dal web server e la invia al server web stesso;

6. Il web server riceve la chiave simmetrica criptata con la chiave pubblica, che pocanzi ha inviato al client e disponendo della sua specifica chiave privata la decripta;

7. A questo punto il client e il server hanno a disposizione la chiave simmetrica a 128 Bit in chiaro tramite la quale criptare e decriptare i dati;

8. Inizia la connessione in SSL e può iniziare la trasmissione sicura.

L'unica annotazione da effettuare è relativa al Web Server; il protocollo SSL, come da figura precedente, lavora su una porta differente da quella http standard (Porta 8080) ma di solito sulla porta 8443. Quindi è importante

ricordare che nei server web in cui sono disponibili servizi SSL occorre abilitare l'accesso del traffico in entrata veicolato sulla porta 8443, altrimenti non sarà possibile effettuare lo *switch* tra le due connessioni.

- ***Utilizzo Di Struts Per Effettuare Comunicazioni Sicure***

Nella realizzazione del framework Struts non è stata prevista nessuna classe in particolare per gestire le comunicazioni sicure. E' però stato creato, da un team gestito da S. Ditlinger, un plug-in denominato SSLEXT (SSL Extension to Struts) [45] che permette di gestire tale connessione in maniera dichiarativa.

E' possibile prelevare da questo sito una libreria, in cui vengono forniti:

- la classe plug-in: *SecurePlugIn*;
- un RequestProcessor personalizzato: *SecureRequestProcessor*;
- una classe action-mapping personalizzata: *SecureActionMapping*;
- una libreria di custom tag JSP: *sslext.tld*.

Prelevata tale libreria e inserita all'interno del progetto bisogna effettuare poche e banali operazioni.

All'interno del file *struts-config.xml* si devono apportare le seguenti modifiche:

1. aggiungere l'attributo type nel tag <action-mappings> nel seguente modo:

```
<action-mappings type="org.apache.struts.config.SecureActionConfig">
```

2. aggiungere il RequestProcessor personalizzato nell'elemento controller:

```
<controller
ProcessorClass="org.apache.struts.action.SecureRequestProcessor"/>
```

3. aggiungere la dichiarazione del plug-in:

```
<plug-in className="org.apache.struts.action.SecurePlugIn">
        <set-property property="httpPort" value="8080"/>
        <set-property property="httpsPort" value="8443"/>
        <set-property property="enable" value="true"/>
        <set-property property="addSession" value="true"/>
</plug-in>
```

4. aggiungere per ogni action la proprietà "secure = true" nel caso in cui deve essere acceduta usando il protocollo https oppure "secure = false" per il protocollo http;

```
<set-property property="secure" value="true"/>
```

5. se si vuole che una pagina JSP sia acceduta direttamente in modalità sicura bisogna innanzitutto definire la libreria di tag *sslext.tld* all'interno del file di configurazione *web.xml* e all'interno delle pagine interessate introdurre:

<%@ taglib uri="/WEB-INF/tlds/sslext.tld" prefix="sslext"%>

<sslext:pageScheme secure="false"/>

Nell'applicazione realizzata non è possibile accedere direttamente alle pagine jsp del modulo di amministrazione, quindi è possibile saltare l'ultimo step.

Tutte le action del modulo amministrazione vengono eseguite con il protocollo https, tranne, ovviamente, quella di logout e quella di login. In figura 5.21 è riportato il funzionamento dell'applicazione quando viene effettuato il login, evidenziando il cambio di protocollo.

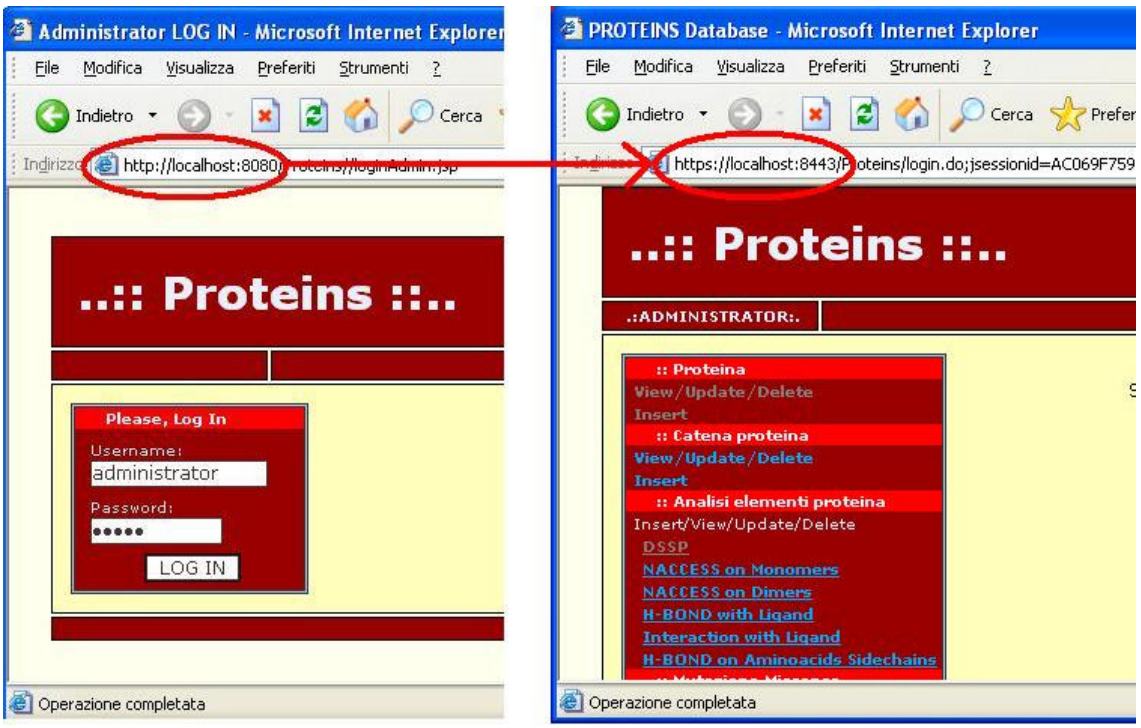


Figura 5.21 - Conseguenza dell'operazione di login, passaggio da http a https

L'operazione opposta accade quando, da una qualunque pagina del modulo Amministrazione, si clicca sul pulsante che realizza il Logout dalle pagine di Amministrazione.

Ovviamente a tutto ciò va premesso che è stato abilitato il Web Server Tomcat per supportare connessioni sulla porta 8443 ed è, inoltre, stato creato un certificato privato per la crittografazione dei dati trasmessi, tramite il comando java *keytool*.

In base a tale implementazione l'amministratore, quando effettua il login, viene informato che il certificato non è stato rilasciato da una Certification Authority(CA) riconosciuta, tramite un messaggio in cui viene richiesta l'accettazione, o meno, della connessione. Tale particolarità è stata indicata nella documentazione rilasciata all'amministratore del sistema, pertanto questi è a conoscenza di dover accettare la connessione. Ovviamente questa scelta è stata obbligata, a causa della mancanza di fondi per l'acquisizione di un certificato rilasciato da una CA riconosciuta.

5.3.5 Consultazione delle Informazioni

L'operazione più importante dell'applicazione realizzata, relativamente al modulo pubblico, è sicuramente la modalità di selezione delle informazioni da consultare. Queste informazioni devono essere relative alla proteina originaria (o Wild Type) oppure alle mutazioni conosciute di tale proteina.

Nella figura 5.22 viene visualizzata la pagina tramite la quale selezionare le informazioni da analizzare:

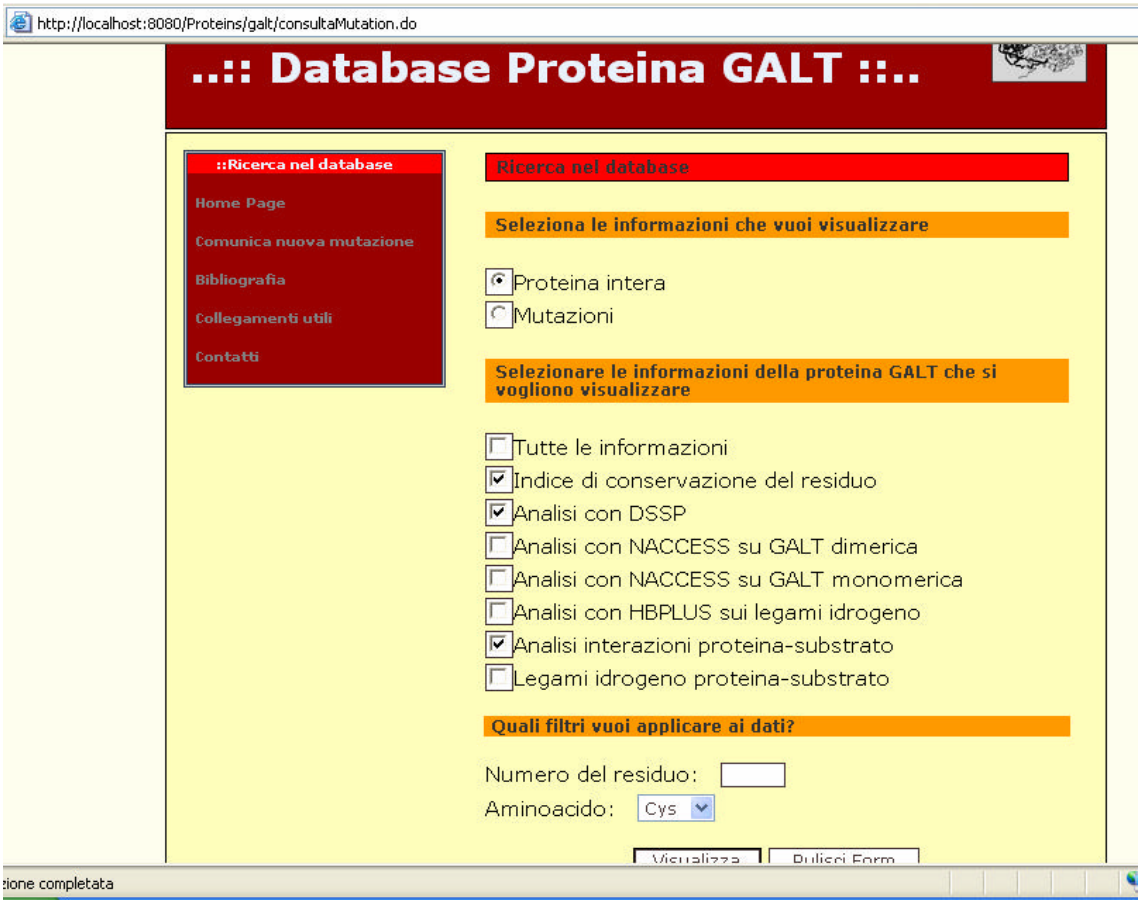


Figura 5.22 – Pagina in cui vengono selezionate le informazioni che si vogliono consultare

Come si può notare, è possibile selezionare le informazioni sulla Proteina intera o sulle mutazioni, e relativamente ad esse le informazioni che si desiderano consultare con la possibilità di restringere la ricerca ad un particolare numero di residuo o ad un particolare aminoacido. La scelta di introdurre questi possibili filtri di ricerca è dovuta alle notevoli dimensioni che può assumere una catena di una proteina. Il risultato di tale selezione viene presentato nella figura 5.23:

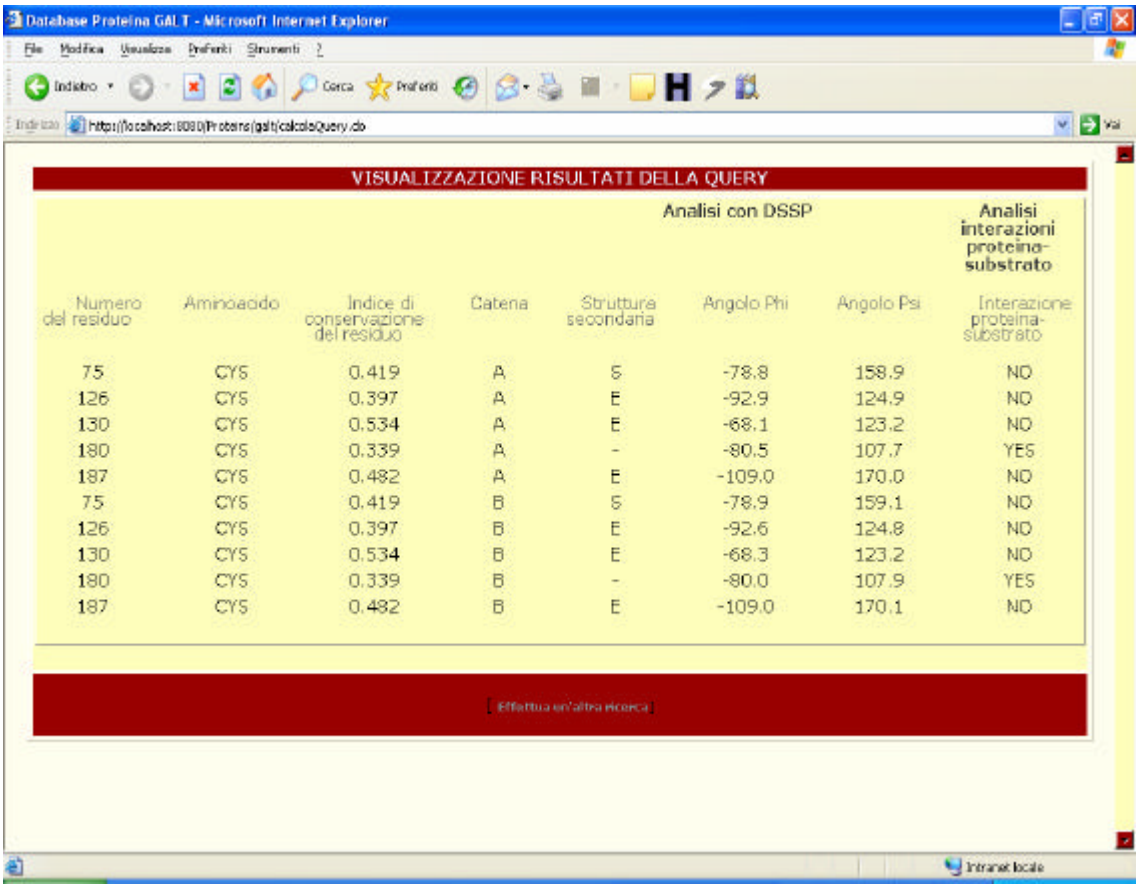


Figura 5.23 – Risultati derivanti dalla selezione precedente

Per completezza visualizziamo la schermata che viene caricata se, invece di selezionare la proteina intera, viene selezionata la consultazione sui dati delle mutazioni (Fig. 5.24):

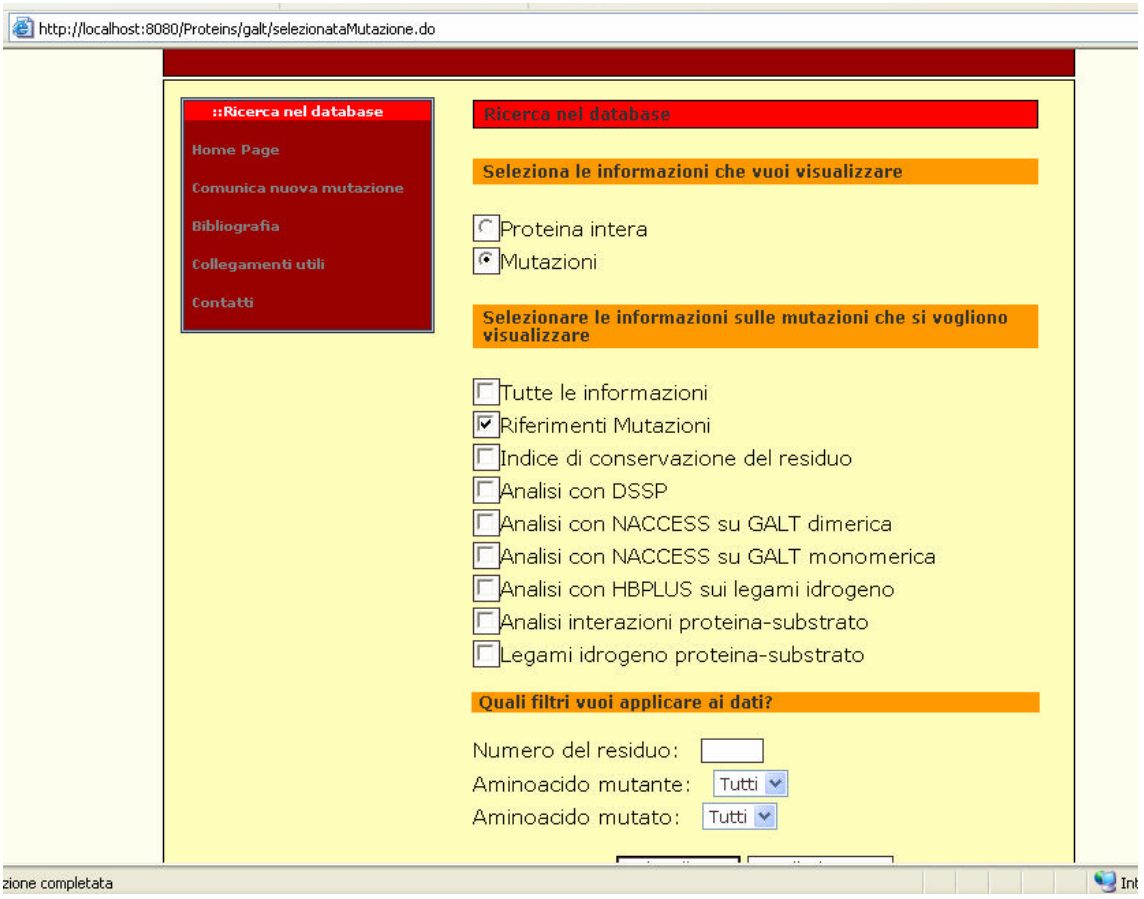


Figura 5.24 – Pagina in cui vengono selezionate le informazioni sulle mutazioni da consultare

Si può notare che, oltre a tutte le voci già presenti prima, viene introdotta la voce "Riferimenti mutazioni", ed inoltre la possibilità di filtrare i risultati sia in base all'Aminoacido "Mutante" che al "Mutato".

Conclusioni

Giunti al termine del lavoro è possibile affermare che tutti gli obiettivi, fissati all'inizio nel primo capitolo, sono stati raggiunti. L'applicazione è perfettamente funzionante e tutte le richieste pervenute sono state soddisfatte. Naturalmente si attendono i giudizi di coloro che utilizzeranno il software, al fine di poter individuare e risolvere eventuali errori non riscontrati finora.

Per quanto riguarda gli sviluppi futuri è possibile dire che, allo stato attuale, nel database, e quindi anche nella parte pubblica dell'applicazione, sono presenti esclusivamente i dati relativi alla proteina GALT. Pertanto devono essere introdotti i dati relativi ad altre proteine di interesse. Tale operazione è facilmente eseguibile dall'Amministratore del Sistema.

Altro possibile sviluppo futuro riguarda l'aggiunta di testi e messaggi in altre lingue. Anche questo aspetto è facilmente realizzabile tramite l'utilizzo di altri resource bundle, il tutto senza dover modificare niente.

Nel corso della realizzazione dell'applicazione, è stato presentato il framework Struts, cercando di illustrare le caratteristiche basilari e quelle ritenute più utili per chi opera nello sviluppo di software per il Web.

Nella letteratura specializzata, nei forum e nei vari siti consultati, Struts viene presentato come un "potente" strumento per costruire applicazioni web di grandi

dimensioni e di alta qualità con una forte riduzione dei tempi di sviluppo e quindi con notevoli vantaggi economici. Ciò inizialmente aveva creato molta curiosità, e forse è stato proprio questo il motivo che ci ha portato alla sua scelta. Ma dopo il suo utilizzo non è possibile far altro che approvare i giudizi di queste persone altamente esperte.

Struts si è rivelato uno strumento molto flessibile, la sua capacità di realizzare le operazioni, anche quelle apparentemente più difficili, in modo dichiarativo è stato di grande aiuto durante tutto il procedimento di implementazione. Ovviamente Struts è un framework in continua evoluzione, data la sua capacità di utilizzare framework esterni, e pertanto non è possibile individuare e trattare tutte le sue caratteristiche. Molto interessante potrebbe essere l'utilizzo del framework Jakarta Commons Logging, che fornisce una serie di strumenti per effettuare il logging delle applicazioni Web, oppure il framework JSF (Java Server Faces), nato a partire da Struts, con lo scopo di fornire allo sviluppatore Web degli utili elementi visuali, e moltissimi altri framework.

APPENDICE A

CREAZIONE ED INIZIALIZZAZIONE DATABASE

Questa appendice illustra la modalità di creazione della struttura e la procedura utilizzata per l'inizializzazione del database

A.1 Creazione Database

L'applicazione utilizza una base di dati denominata ***proteins***. Il DBMS utilizzato è PostgreSQL. Per la creazione della struttura del database è possibile far generare al CASE PowerDesigner uno script per il particolare DBMS. Viene riportato qui di seguito tale script:

```
/*==============================================================*/
/* DBMS name:      PostgreSQL 7.3                               */
/* Created on:     20/11/2005 13.12.55                          */
/*==============================================================*/

drop index AMINOACIDI_PK;

drop index CATENA_PROTEINA_PK;

drop index COMPOSIZIONE_FK;

drop index DETTAGLI_PROTEINA_FK;

drop index CODIFICA_FK;

drop index CODONS_PK;

drop index ANALISI2_FK;

drop index DSSP_PK;

drop index ANALISI5_FK;

drop index H_BONDS_SIDECHAINS_PK;

drop index ANALISI_FK;

drop index H_BOND_CON_LIGANDO_PK;

drop index ANALISI6_FK;

drop index INTERAZIONE_CON_LIGANDO_PK;

drop index LOGINADMIN_PK;

drop index AMINOACIDO_MUTANTE_FK;

drop index MUTANTE_FK;

drop index MUTATO_FK;

drop index MUTAZIONI_MISSENSE_PK;

drop index ANALISI3_FK;
```

```
drop index NACCESS_DIMER_PK;

drop index ANALISI4_FK;

drop index NACCESS_MONOMERS_PK;

drop index PROTEINE_PK;

drop table AMINOACIDI;

drop table CATENA_PROTEINA;

drop table CODONS;

drop table DSSP;

drop table H_BONDS_SIDECHAINS;

drop table H_BOND_CON_LIGANDO;

drop table INTERAZIONE_CON_LIGANDO;

drop table LOGINADMIN;

drop table MUTAZIONI_MISSENSE;

drop table NACCESS_DIMER;

drop table NACCESS_MONOMERS;

drop table PROTEINE;

/*==============================================================*/
/* Table: AMINOACIDI                                            */
/*==============================================================*/
create table AMINOACIDI (
NOME_3_CHAR          CHAR(3)              not null,
NOME_1_CHAR          CHAR(1)              not null,
NOME_INTERO          TEXT                 not null
);

/*==============================================================*/
/* Index: AMINOACIDI_PK                                         */
/*==============================================================*/
create unique index AMINOACIDI_PK on AMINOACIDI (
NOME_3_CHAR
);

/*==============================================================*/
/* Table: CATENA_PROTEINA                                       */
/*==============================================================*/
create table CATENA_PROTEINA (
COD_PROT             VARCHAR(15)          not null,
NUMERO_RESIDUO       NUMERIC(5,0)         not null,
NOME_3_CHAR          CHAR(3)              not null,
CONSERVATION_SCORE   DECIMAL(4,3)         null
);

/*==============================================================*/
/* Index: CATENA_PROTEINA_PK                                    */
/*==============================================================*/
create unique index CATENA_PROTEINA_PK on CATENA_PROTEINA (
COD_PROT,
NUMERO_RESIDUO
```

```
);

/*==============================================================*/
/* Index: DETTAGLI_PROTEINA_FK                                  */
/*==============================================================*/
create  index DETTAGLI_PROTEINA_FK on CATENA_PROTEINA (
COD_PROT
);

/*==============================================================*/
/* Index: COMPOSIZIONE_FK                                       */
/*==============================================================*/
create  index COMPOSIZIONE_FK on CATENA_PROTEINA (
NOME_3_CHAR
);

/*==============================================================*/
/* Table: CODONS                                                */
/*==============================================================*/
create table CODONS (
NOME_3_CHAR          CHAR(3)              not null,
TRIPLETTA_NUCLEOTIDI CHAR(3)              not null
);

/*==============================================================*/
/* Index: CODONS_PK                                             */
/*==============================================================*/
create unique index CODONS_PK on CODONS (
NOME_3_CHAR,
TRIPLETTA_NUCLEOTIDI
);

/*==============================================================*/
/* Index: CODIFICA_FK                                           */
/*==============================================================*/
create  index CODIFICA_FK on CODONS (
NOME_3_CHAR
);

/*==============================================================*/
/* Table: DSSP                                                  */
/*==============================================================*/
create table DSSP (
COD_PROT             VARCHAR(15)          not null,
NUMERO_RESIDUO       NUMERIC(5,0)         not null,
CATENA               CHAR(1)              not null,
STRUTTURA_SECONDARIA CHAR(1)              not null default '-',
PHI_ANGLE            NUMERIC(3)           not null,
PSI_ANGLE            NUMERIC(3)           not null
);

/*==============================================================*/
/* Index: DSSP_PK                                               */
/*==============================================================*/
create unique index DSSP_PK on DSSP (
COD_PROT,
NUMERO_RESIDUO,
CATENA
);

/*==============================================================*/
/* Index: ANALISI2_FK                                           */
/*==============================================================*/
create  index ANALISI2_FK on DSSP (
```

```
COD_PROT,
NUMERO_RESIDUO
);

/*==============================================================*/
/* Table: H_BONDS_SIDECHAINS                                    */
/*==============================================================*/
create table H_BONDS_SIDECHAINS (
COD_PROT             VARCHAR(15)          not null,
NUMERO_RESIDUO       NUMERIC(5,0)         not null,
CATENA               CHAR(1)              not null,
DONORSIDECHAIN       BOOL                 not null,
ACCEPTORSIDECHAIN    BOOL                 not null,
DONORBACKBONE        BOOL                 not null,
ACCEPTORBACKBONE     BOOL                 not null
);

/*==============================================================*/
/* Index: H_BONDS_SIDECHAINS_PK                                 */
/*==============================================================*/
create unique index H_BONDS_SIDECHAINS_PK on H_BONDS_SIDECHAINS (
COD_PROT,
NUMERO_RESIDUO,
CATENA
);

/*==============================================================*/
/* Index: ANALISI5_FK                                           */
/*==============================================================*/
create  index ANALISI5_FK on H_BONDS_SIDECHAINS (
COD_PROT,
NUMERO_RESIDUO
);

/*==============================================================*/
/* Table: H_BOND_CON_LIGANDO                                    */
/*==============================================================*/
create table H_BOND_CON_LIGANDO (
COD_PROT             VARCHAR(15)          not null,
NUMERO_RESIDUO       NUMERIC(5,0)         not null,
CATENA               CHAR(1)              not null,
DONATORE             BOOL                 not null,
ACCETTORE            BOOL                 not null
);

/*==============================================================*/
/* Index: H_BOND_CON_LIGANDO_PK                                 */
/*==============================================================*/
create unique index H_BOND_CON_LIGANDO_PK on H_BOND_CON_LIGANDO (
COD_PROT,
NUMERO_RESIDUO,
CATENA
);

/*==============================================================*/
/* Index: ANALISI_FK                                            */
/*==============================================================*/
create  index ANALISI_FK on H_BOND_CON_LIGANDO (
COD_PROT,
NUMERO_RESIDUO
);

/*==============================================================*/
/* Table: INTERAZIONE_CON_LIGANDO                               */
```

```
/*==============================================================*/
create table INTERAZIONE_CON_LIGANDO (
COD_PROT             VARCHAR(15)          not null,
NUMERO_RESIDUO       NUMERIC(5,0)         not null,
CATENA               CHAR(1)              not null,
INTERACTION          BOOL                 not null
);

/*==============================================================*/
/* Index: INTERAZIONE_CON_LIGANDO_PK                            */
/*==============================================================*/
create unique index INTERAZIONE_CON_LIGANDO_PK on INTERAZIONE_CON_LIGANDO (
COD_PROT,
NUMERO_RESIDUO,
CATENA
);

/*==============================================================*/
/* Index: ANALISI6_FK                                           */
/*==============================================================*/
create  index ANALISI6_FK on INTERAZIONE_CON_LIGANDO (
COD_PROT,
NUMERO_RESIDUO
);

/*==============================================================*/
/* Table: LOGINADMIN                                            */
/*==============================================================*/
create table LOGINADMIN (
USERNAME             VARCHAR(15)          not null,
PASSWORD             VARCHAR(15)          null,
EMAIL                VARCHAR(30)          null
);

/*==============================================================*/
/* Index: LOGINADMIN_PK                                         */
/*==============================================================*/
create unique index LOGINADMIN_PK on LOGINADMIN (
USERNAME
);

/*==============================================================*/
/* Table: MUTAZIONI_MISSENSE                                    */
/*==============================================================*/
create table MUTAZIONI_MISSENSE (
NOME_MUTAZIONE       VARCHAR(5)           not null,
COD_NOME_3_CHAR      CHAR(3)              not null,
TRIPLETTA_NUCLEOTIDI CHAR(3)              not null,
COD_PROT             VARCHAR(15)          not null,
NUMERO_RESIDUO       NUMERIC(5,0)         not null,
NOME_3_CHAR          CHAR(3)              not null,
COD_TRIPLETTA_NUCLEOTIDI CHAR(3)              not null,
RIFERIMENTI          TEXT                 null
);

/*==============================================================*/
/* Index: MUTAZIONI_MISSENSE_PK                                 */
/*==============================================================*/
create unique index MUTAZIONI_MISSENSE_PK on MUTAZIONI_MISSENSE (
NOME_MUTAZIONE
);

/*==============================================================*/
/* Index: AMINOACIDO_MUTANTE_FK                                 */
```

```
/*==============================================================*/
create  index AMINOACIDO_MUTANTE_FK on MUTAZIONI_MISSENSE (
COD_PROT,
NUMERO_RESIDUO
);

/*==============================================================*/
/* Index: MUTATO_FK                                             */
/*==============================================================*/
create  index MUTATO_FK on MUTAZIONI_MISSENSE (
NOME_3_CHAR,
COD_TRIPLETTA_NUCLEOTIDI
);

/*==============================================================*/
/* Index: MUTANTE_FK                                            */
/*==============================================================*/
create  index MUTANTE_FK on MUTAZIONI_MISSENSE (
COD_NOME_3_CHAR,
TRIPLETTA_NUCLEOTIDI
);

/*==============================================================*/
/* Table: NACCESS_DIMER                                         */
/*==============================================================*/
create table NACCESS_DIMER (
COD_PROT             VARCHAR(15)          not null,
NUMERO_RESIDUO       NUMERIC(5,0)         not null,
CATENA               CHAR(1)              not null,
ABSDIMER             DECIMAL(5,2)         not null,
RELDIMER             DECIMAL(5,2)         not null
);

/*==============================================================*/
/* Index: NACCESS_DIMER_PK                                      */
/*==============================================================*/
create unique index NACCESS_DIMER_PK on NACCESS_DIMER (
COD_PROT,
NUMERO_RESIDUO,
CATENA
);

/*==============================================================*/
/* Index: ANALISI3_FK                                           */
/*==============================================================*/
create  index ANALISI3_FK on NACCESS_DIMER (
COD_PROT,
NUMERO_RESIDUO
);

/*==============================================================*/
/* Table: NACCESS_MONOMERS                                      */
/*==============================================================*/
create table NACCESS_MONOMERS (
COD_PROT             VARCHAR(15)          not null,
NUMERO_RESIDUO       NUMERIC(5,0)         not null,
CATENA               CHAR(1)              not null,
ABSMONOMER           DECIMAL(5,2)         not null,
RELMONOMER           DECIMAL(5,2)         not null
);

/*==============================================================*/
/* Index: NACCESS_MONOMERS_PK                                   */
/*==============================================================*/
```

```
create unique index NACCESS_MONOMERS_PK on NACCESS_MONOMERS (
COD_PROT,
NUMERO_RESIDUO,
CATENA
);

/*==============================================================*/
/* Index: ANALISI4_FK                                           */
/*==============================================================*/
create  index ANALISI4_FK on NACCESS_MONOMERS (
COD_PROT,
NUMERO_RESIDUO
);

/*==============================================================*/
/* Table: PROTEINE                                              */
/*==============================================================*/
create table PROTEINE (
COD_PROT             VARCHAR(15)          not null,
DESCRIZIONE          VARCHAR(100)         not null
);

/*==============================================================*/
/* Index: PROTEINE_PK                                           */
/*==============================================================*/
create unique index PROTEINE_PK on PROTEINE (
COD_PROT
);

alter table CATENA_PROTEINA
   add constraint FK_CATENA_P_COMPOSIZI_AMINOACI foreign key (NOME_3_CHAR)
      references AMINOACIDI (NOME_3_CHAR)
      on delete restrict on update restrict;

alter table CATENA_PROTEINA
   add constraint FK_CATENA_P_DETTAGLI__PROTEINE foreign key (COD_PROT)
      references PROTEINE (COD_PROT)
      on delete restrict on update restrict;

alter table CODONS
   add constraint FK_CODONS_CODIFICA_AMINOACI foreign key (NOME_3_CHAR)
      references AMINOACIDI (NOME_3_CHAR)
      on delete restrict on update restrict;

alter table DSSP
   add constraint FK_DSSP_ANALISI2_CATENA_P foreign key (COD_PROT,
NUMERO_RESIDUO)
      references CATENA_PROTEINA (COD_PROT, NUMERO_RESIDUO)
      on delete restrict on update restrict;

alter table H_BONDS_SIDECHAINS
   add constraint FK_H_BONDS__ANALISI5_CATENA_P foreign key (COD_PROT,
NUMERO_RESIDUO)
      references CATENA_PROTEINA (COD_PROT, NUMERO_RESIDUO)
      on delete restrict on update restrict;

alter table H_BOND_CON_LIGANDO
   add constraint FK_H_BOND_C_ANALISI_CATENA_P foreign key (COD_PROT,
NUMERO_RESIDUO)
      references CATENA_PROTEINA (COD_PROT, NUMERO_RESIDUO)
      on delete restrict on update restrict;

alter table INTERAZIONE_CON_LIGANDO
```

```
   add constraint FK_INTERAZI_ANALISI6_CATENA_P foreign key (COD_PROT,
NUMERO_RESIDUO)
      references CATENA_PROTEINA (COD_PROT, NUMERO_RESIDUO)
      on delete restrict on update restrict;

alter table MUTAZIONI_MISSENSE
   add constraint FK_MUTAZION_AMINOACID_CATENA_P foreign key (COD_PROT,
NUMERO_RESIDUO)
      references CATENA_PROTEINA (COD_PROT, NUMERO_RESIDUO)
      on delete restrict on update restrict;

alter table MUTAZIONI_MISSENSE
   add constraint FK_MUTAZION_MUTANTE_CODONS foreign key (COD_NOME_3_CHAR,
TRIPLETTA_NUCLEOTIDI)
      references CODONS (NOME_3_CHAR, TRIPLETTA_NUCLEOTIDI)
      on delete restrict on update restrict;

alter table MUTAZIONI_MISSENSE
   add constraint FK_MUTAZION_MUTATO_CODONS foreign key (NOME_3_CHAR,
COD_TRIPLETTA_NUCLEOTIDI)
      references CODONS (NOME_3_CHAR, TRIPLETTA_NUCLEOTIDI)
      on delete restrict on update restrict;

alter table NACCESS_DIMER
   add constraint FK_NACCESS__ANALISI3_CATENA_P foreign key (COD_PROT,
NUMERO_RESIDUO)
      references CATENA_PROTEINA (COD_PROT, NUMERO_RESIDUO)
      on delete restrict on update restrict;

alter table NACCESS_MONOMERS
   add constraint FK_NACCESS__ANALISI4_CATENA_P foreign key (COD_PROT,
NUMERO_RESIDUO)
      references CATENA_PROTEINA (COD_PROT, NUMERO_RESIDUO)
```

on delete restrict on update restrict

A.2 Inizializzazione Database

I dati a nostra disposizione sono relativi alla proteina GALT. Essi sono prevalentemente disponibili in file di Microsoft Excel, con estensione *.xls*. Per tale motivo, per l'inizializzazione del database *proteins* si è utilizzata una metodologia in 4 passi, con l'utilizzo di Microsoft Access:

1. è stata creata la struttura del database in Microsoft Access, con l'utilizzo di uno script generato con il CASE PowerDesigner;

2. sono stati caricati i dati in tabelle di Access, direttamente dai file Excel;

3. è stato creato un file con estensione *.csv* per ogni tabella, i cui dati devono essere trasferiti;

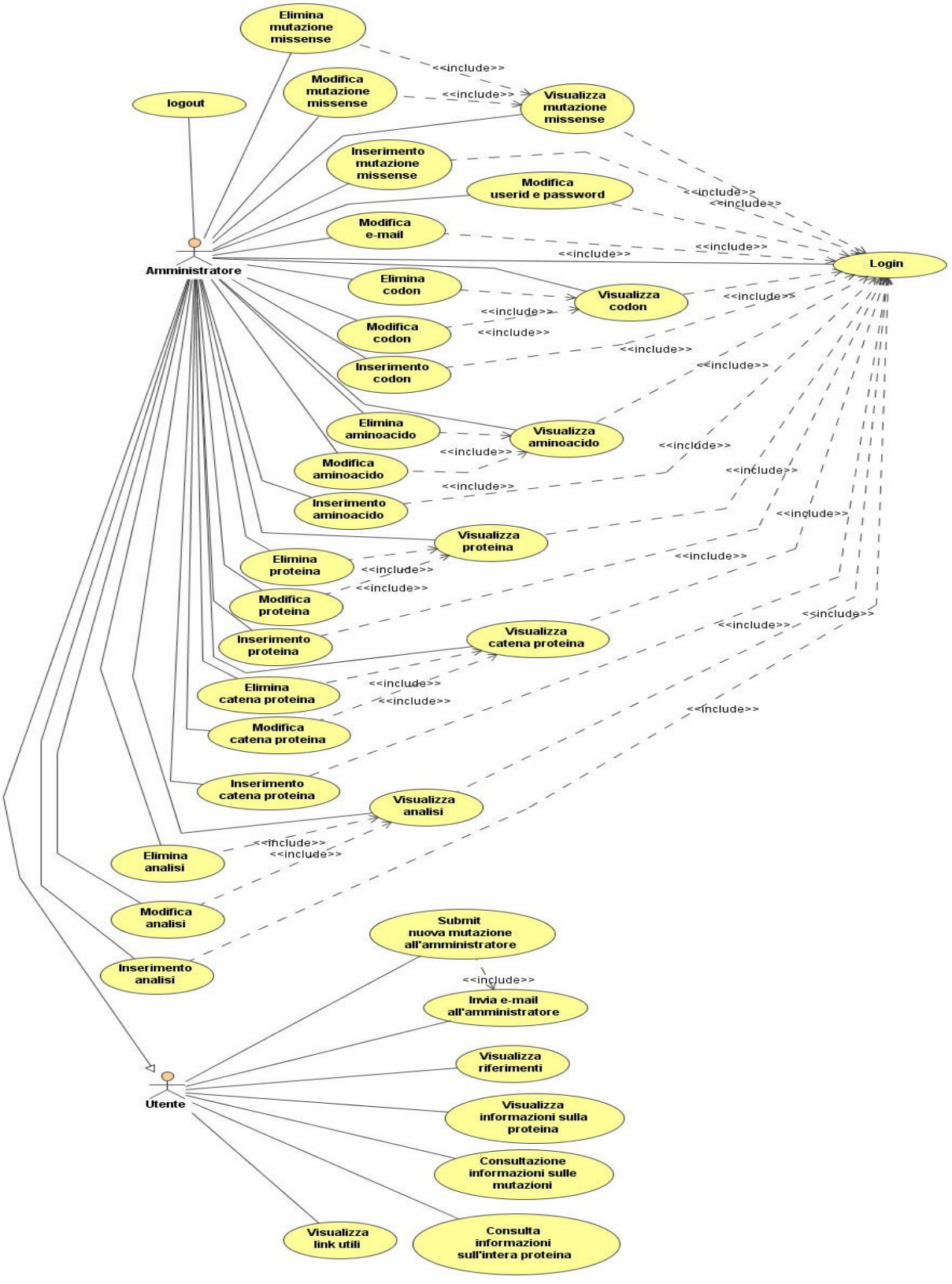
4. mediante l'utilizzo del comando COPY di PostgreSQL è stato trasferito il contenuto dei file *.csv* all'interno della base di dati.

APPENDICE B

FASE DI PROGETTAZIONE: I DIAGRAMMI DELL'APPLICAZIONE

In questa appendice vengono riportati i diagrammi realizzati durante la fase di progettazione dell'applicazione. Vengono inoltre riportati i Casi d'Uso in forma testuale.

B.1 Use Cases Diagram



B.2 Use Cases in Forma Testuale

Caso d'uso: Login

Scenario 1

1. L'utente decide di procedere al Login nel sistema
2. L'amministratore inserisce Username e Password
3. Il sistema verifica che lo Username e la Password inseriti siano quelli dell'amministratore
4. Il sistema riconosce l'amministratore e abilita le funzionalità di amministrazione

Scenario 2

1. L'utente decide di procedere al Login nel sistema
2. L'amministratore inserisce Username e Password
3. Il sistema verifica che lo Username e la Password inseriti siano quelli dell'amministratore
4. Il sistema informa l'amministratore che la Username è errata
5. L'amministratore esce dal sistema o procede ad un nuovo login

Scenario 3

1. L'utente decide di procedere al Login nel sistema
2. L'amministratore inserisce Username e Password
3. Il sistema verifica che lo Username e la Password inseriti siano quelli dell'amministratore
4. Il sistema informa l'amministratore che la Password è errata
5. L'amministratore esce dal sistema o procede ad un nuovo login

Caso d'uso: modifica username/password di amministrazione

Scenario 1

1. **Include Login**
2. L'amministratore inserisce una nuova Username e una nuova password di accesso
3. Il sistema verifica la sua validità
4. Il sistema registra la nuova password

Scenario 2

1. **Include Login**
2. L'amministratore inserisce una nuova Username e una nuova password di accesso
3. Il sistema verifica la sua validità
4. Il sistema informa che la nuova Username non è valida
5. Il sistema effettua il Logout dell'amministratore

Scenario 3

1. **Include Login**
2. L'amministratore inserisce una nuova Username e una nuova password di accesso
3. Il sistema verifica la sua validità
4. Il sistema informa che la nuova Password non è valida
5. Il sistema effettua il Logout dell'amministratore

Caso d'uso: modifica email amministratore

Scenario 1

1. **Include Login**
2. L'amministratore inserisce una nuova email

3. Il sistema verifica la sua validità
4. Il sistema registra la nuova email

Scenario 2

1. **Include Login**
2. L'amministratore inserisce una nuova email
3. Il sistema verifica la sua validità
4. Il sistema informa che la nuova email non è valida
5. L'amministratore inserisce una nuova email o esce dal sistema

Caso d'uso: Inserimento codon

Scenario 1

1. **Include Login**
2. L'amministratore inserisce i dati relativi al codon
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che il codon sia lungo 3 caratteri
5. Il sistema verifica che il codon non sia già presente nella banca dati
6. Il sistema verifica che l'aminoacido associato sia presente nella banca dati
7. Il sistema registra i dati nella banca dati

Scenario 2

1. **Include Login**
2. L'amministratore inserisce i dati relativi al codon
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema informa che i dati obbligatori non sono stati forniti
5. L'amministratore provvede a fornire i dati o interrompe l'inserimento

Scenario 3

1. **Include Login**
2. L'amministratore inserisce i dati relativi al codon
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che il codon sia lungo 3 caratteri
5. Il sistema informa che il codon non è della lunghezza prevista
6. L'amministratore provvede a fornire i dati o interrompe l'inserimento

Scenario 4

1. **Include Login**
2. L'amministratore inserisce i dati relativi al codon
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che il codon sia lungo 3 caratteri
5. Il sistema verifica che il codon non sia già presente nella banca dati
6. Il sistema informa che il codon è già presente
7. L'amministratore provvede a modificare i dati o interrompe l'inserimento

Scenario 5

1. **Include Login**
2. L'amministratore inserisce i dati relativi al codon
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che il codon non sia già presente nella banca dati
5. Il sistema verifica che l'aminoacido associato sia presente nella banca dati
6. Il sistema informa che l'aminoacido associato non è presente nella banca dati

7. L'amministratore provvede a modificare i dati dell'aminoacido o interrompe l'inserimento

Caso d'uso: Visualizzazione codon

Scenario 1

1. **Include Login**
2. L'amministratore richiede l'elenco dei codon presenti nella banca dati
3. Il sistema verifica che siano presenti codon nella banca dati
4. Il sistema fornisce l'elenco dei codon

Scenario 2

1. **Include Login**
2. L'amministratore richiede l'elenco dei codon presenti nella banca dati
3. Il sistema verifica che siano presenti codon nella banca dati
4. Il sistema informa l'Amministratore che non sono presenti codon nella banca dati
5. L'amministratore annulla l'operazione

Caso d'uso: Modifica codon

Scenario 1

1. **include Visualizzazione codon**
2. L'amministratore inserisce il valore codon da modificare
3. Il sistema verifica se il valore del codon è valido
4. L'amministratore modifica i dati relativi al codon
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica se il valore del codon inserito è di 3 caratteri
7. Il sistema verifica che il codon modificato non sia già presente nella banca dati
8. Il sistema verifica che il nome aminoacido modificato sia presente nella banca dati
9. Il sistema verifica se ci sono Mutazioni in cui è presente il codon modificato e in caso positivo provvede a fare l'aggiornamento in cascata
10. Il sistema provvede a registrare i nuovi dati

Scenario 2

1. **include Visualizzazione codon**
2. L'amministratore inserisce il valore del codon da modificare
3. Il sistema verifica se il valore del codon è valido
4. Il sistema informa che al valore richiesto non corrisponde nessun codon
5. L'amministratore inserisce un nuovo valore del codon o annulla l'operazione

Scenario 3

1. **include Visualizzazione codon**
2. L'amministratore inserisce il valore del codon da modificare
3. Il sistema verifica se il valore inserito è valido
4. L'amministratore modifica i dati relativi al codon
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
7. L'amministratore provvede a inserire i dati obbligatori mancanti o annulla le modifiche

Scenario 4

1. **include Visualizzazione codon**
2. L'amministratore inserisce il valore codon da modificare
3. Il sistema verifica se il valore del codon è valido

4. L'amministratore modifica i dati relativi al codon
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica se il valore del codon inserito è di 3 caratteri
7. Il sistema informa che il codon inserito non è della lunghezza stabilita
8. L'amministratore provvede a ripetere l'inerimento o annulla l'operazione

Scenario 5

1. **include Visualizzazione codon**
2. L'amministratore inserisce il valore codon da modificare
3. Il sistema verifica se il valore del codon è valido
4. L'amministratore modifica i dati relativi al codon
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica se il valore del codon inserito è di 3 caratteri
7. Il sistema verifica che il codon modificato non sia già presente nella banca dati
8. Il sistema informa l'amministratore che il codon inserito è già presente nella banca dati
9. L'amministratore provvede a modificare i dati o annulla le modifiche

Scenario 6

1. **include Visualizzazione codon**
2. L'amministratore inserisce il valore codon da modificare
3. Il sistema verifica se il valore del codon è valido
4. L'amministratore modifica i dati relativi al codon
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica se il valore del codon inserito è di 3 caratteri
7. Il sistema verifica che il codon modificato non sia già presente nella banca dati
8. Il sistema verifica che l'aminoacido associato modificato sia presente nella banca dati
9. Il sistema informa l'amministratore che l'aminoacido modificato non è presente nella banca dati
10. L'amministratore provvede a modificare i dati o annulla le modifiche

Caso d'uso: Eliminazione codon

Scenario 1

1. **include Visualizzazione codon**
2. L'amministratore inserisce il valore del codon da eliminare
3. Il sistema verifica se il valore inserito è valido
4. Il sistema verifica che non ci siano Mutazioni Missense associate
5. Il sistema elimina il codon

Scenario 2

1. **include Visualizzazione codon**
2. L'amministratore inserisce il valore del codon da eliminare
3. Il sistema verifica se il valore inserito è valido
4. Il sistema informa l'amministratore che il valore non è valido
5. L'amministratore inserisce un nuovo valore o annulla l'operazione

Scenario 3

1. **include Visualizzazione codon**
2. L'amministratore inserisce il valore del codon da eliminare
3. Il sistema verifica se il valore inserito è valido
4. Il sistema verifica che non ci siano Mutazioni Missense associate

5. Il sistema informa che il valore del codon è associato a qualche mutazione e quindi non è possibile cancellarlo
6. L'amministratore provvede a modificare il valore del codon da eliminare o annulla l'operazione

Caso d'uso: Inserimento aminoacido

Scenario 1

1. **include Login**
2. L'amministratore inserisce i dati relativi all'aminoacido
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che l'aminoacido inserito sia valido
5. Il sistema verifica che l'aminoacido non sia già presente nella banca dati
6. Il sistema verifica che il nome di 1 carattere e il nome a 3 caratteri non siano più lunghi/brevi del previsto
7. Il sistema registra i dati nella banca dati

Scenario 2

1. **include Login**
2. L'amministratore inserisce i dati relativi all'aminoacido
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
5. L'amministratore inserisce i dati mancanti o annulla l'inserimento

Scenario 3

1. **include Login**
2. L'amministratore inserisce i dati relativi all'aminoacido
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che l'aminoacido inserito sia valido
5. Il sistema informa l'amministratore che i dati inseriti non sono validi
6. L'amministratore modifica i dati o annulla l'inserimento

Scenario 4

1. **include Login**
2. L'amministratore inserisce i dati relativi all'aminoacido
3. Il sistema verifica che l'aminoacido inserito sia valido
4. Il sistema verifica l'inserimento dei dati obbligatori
5. Il sistema verifica che l'aminoacido non sia già presente nella banca dati
6. Il sistema informa l'amministratore che l'aminoacido è già presente nella banca dati
7. L'amministratore modifica i dati o annulla l'inserimento

Scenario 5

1. **include Login**
2. L'amministratore inserisce i dati relativi all'aminoacido
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che l'aminoacido inserito sia valido
5. Il sistema verifica che l'aminoacido non sia già presente nella banca dati
6. Il sistema verifica che il nome di 1 carattere e il nome a 3 caratteri non siano più lunghi/brevi del previsto
7. Il sistema informa l'amministratore che il nome ad 1 carattere e/o il nome a 3 caratteri inseriti non sono della lunghezza richiesta
8. L'amministratore modifica i dati o annulla l'inserimento

Caso d'uso: Visualizzazione aminoacido

Scenario 1

1. **Include Login**
2. L'amministratore richiede l'elenco degli aminoacidi presenti nella banca dati
3. Il sistema verifica che siano presenti aminoacidi nella banca dati
4. Il sistema fornisce l'elenco degli aminoacidi

Scenario 2

1. **Include Login**
2. L'amministratore richiede l'elenco degli aminoacidi presenti nella banca dati
3. Il sistema verifica che siano presenti aminoacidi nella banca dati
4. Il sistema informa l'Amministratore che non sono presenti aminoacidi nella banca dati
5. L'amministratore annulla l'operazione

Caso d'uso: Modifica aminoacido

Scenario 1

1. **include Visualizzazione aminoacido**
2. L'amministratore inserisce il nome dell'aminoacido da modificare
3. Il sistema verifica se il nome inserito è valido
4. L'amministratore modifica i dati relativi all'aminoacido
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che il nome ad 1 carattere non e il nome a 3 caratteri non siano più lunghi/brevi del previsto
7. Il sistema verifica che il l'aminoacido modificato non sia già presente nella banca dati
8. Il sistema verifica se l'aminoacido modificato è presente nella Lista dei Codons o in qualche Catena della Proteina e in caso affermativo ne effettua l'aggiornamento in cascata
9. Il sistema provvede a registrare i nuovi dati

Scenario 2

1. **include Visualizzazione aminoacido**
2. L'amministratore inserisce l'aminoacido da modificare
3. Il sistema verifica se il nome inserito è valido
4. Il sistema informa l'amministratore che il nome inserito non valido
5. L'amministratore provvede a inserire nuovamente il nome o annulla le modifiche

Scenario 3

1. **include Visualizzazione aminoacido**
2. L'amministratore inserisce il nome dell'aminoacido da modificare
3. Il sistema verifica se il nome inserito è valido
4. L'amministratore modifica i dati relativi all'aminoacido
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
7. L'amministratore provvede ad inserire i dati obbligatori o annulla le modifiche

Scenario 4

1. **include Visualizzazione aminoacido**
2. L'amministratore inserisce il nome dell'aminoacido da modificare
3. Il sistema verifica se il nome inserito è valido
4. L'amministratore modifica i dati relativi all'aminoacido
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti

6. Il sistema verifica che il nome ad 1 carattere non e il nome a 3 caratteri non siano più lunghi/brevi del previsto
7. Il sistema informa che il nome 1 carattere e/o il nome a 3 caratteri non sono delle dimensioni richieste
8. L'amministratore provvede a modificare i valori o annulla l'operazione

Scenario 5

1. **include Visualizzazione aminoacido**
2. L'amministratore inserisce il nome dell'aminoacido da modificare
3. Il sistema verifica se il nome inserito è valido
4. L'amministratore modifica i dati relativi all'aminoacido
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che il nome ad 1 carattere non e il nome a 3 caratteri non siano più lunghi/brevi del previsto
7. Il sistema verifica che il l'aminoacido modificato non sia già presente nella banca dati
8. Il sistema informa l'amministratore che l'aminoacido inserito è già presente nella banca dati
9. L'amministratore provvede a modificare i dati o annulla le modifiche

Caso d'uso: Eliminazione aminoacido

Scenario 1

1. **include Visualizzazione aminoacido**
2. L'amministratore inserisce il nome dell'aminoacido da eliminare
3. Il sistema verifica se il valore inserito è valido
4. Il sistema verifica se l'aminoacido selezionato è associato a qualche codon o a qualche elemento della catena della proteina
5. Il sistema elimina l'aminoacido

Scenario 2

1. **include Visualizzazione aminoacido**
2. L'amministratore inserisce il nome dell'aminoacido da eliminare
3. Il sistema verifica se il valore inserito è valido
4. Il sistema informa l'amministratore che il nome inserito non valido
5. L'amministratore provvede a inserire nuovamente il nome o annulla le modifiche

Scenario 3

1. **include Visualizzazione aminoacido**
2. L'amministratore inserisce il nome dell'aminoacido da eliminare
3. Il sistema verifica se il valore inserito è valido
4. Il sistema verifica se l'aminoacido selezionato è associato a qualche codon o a qualche elemento della catena della proteina
5. Il sistema informa l'amministratore che è impossibile eliminare l'aminoacido in quanto esso è già associato ad almeno un codon e/o a qualche elemento in qualche catena della proteina
6. L'amministratore modifica l'elemento da eliminare o annulla l'eliminazione

Caso d'uso: Inserimento Proteina

Scenario 1

1. **include Login**

2. L'amministratore inserisce i dati relativi a proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che la proteina non sia già presente nella banca dati
5. Il sistema registra i dati nella banca dati

Scenario 2

1. **include Login**
2. L'amministratore inserisce i dati relativi a proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema informa l'amministratore i dati obbligatori non sono stati inseriti
5. L'amministratore inserisce i dati obbligatori o annulla l'inserimento

Scenario 3

1. **include Login**
2. L'amministratore inserisce i dati relativi a proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che la proteina non sia già presente nella banca dati
5. Il sistema informa l'amministratore che la proteina è già presente nella banca dati

Caso d'uso: Visualizza proteine

Scenario 1

1. **include Login**
2. L'amministratore richiede l'elenco delle proteine presenti nella banca dati
3. Il sistema verifica che siano presenti proteine nella banca dati
4. Il sistema fornisce l'elenco delle proteine

Scenario 2

1. **include Login**
2. L'amministratore richiede l'elenco delle proteine presenti nella banca dati
3. Il sistema verifica che siano presenti proteine nella banca dati
4. Il sistema informa l'Amministratore che non sono presenti proteine nella banca dati
5. L'Amministratore annulla l'operazione

Caso d'uso: Modifica proteina

Scenario 1

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da modificare
3. Il sistema verifica se il codice inserito è valido
4. L'amministratore modifica i dati relativi alla proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che la proteina modificata non sia già presente nella banca dati
7. Il sistema verifica se la proteina modificata è presente in qualche Mutazione Missense o in qualche catena della proteina e in caso affermativo provvede all'aggiornamento in cascata
8. Il sistema provvede a registrare i nuovi dati

Scenario 2

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da modificare
3. Il sistema verifica se il codice inserito è valido

4. Il sistema informa l'amministratore che il codice inserito non è valido
5. L'amministratore modifica il codice o annulla le modifiche

Scenario 3

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da modificare
3. Il sistema verifica se il codice inserito è valido
4. L'amministratore modifica i dati relativi alla proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
7. L'amministratore inserisci i dati obbligatori o annulla le modifiche

Scenario 4

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da modificare
3. Il sistema verifica se il codice inserito è valido
4. L'amministratore modifica i dati relativi alla proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che la proteina modificata non sia già presente nella banca dati
7. Il sistema informa l'amministratore che la proteina è già presente nella banca dati
8. L'amministratore modifica il codice della proteina o annulla le modifiche

Caso d'uso: Eliminazione proteina

Scenario 1

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da eliminare
3. Il sistema verifica se il codice inserito è valido
4. Il sistema verifica se alla proteina è già associato almeno un elemento nella catena della proteina o a qualche mutazione missense
5. Il sistema elimina la proteina

Scenario 2

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da eliminare
3. Il sistema verifica se il codice inserito è valido
4. Il sistema informa l'amministratore che il codice inserito non è valido
5. L'amministratore modifica il codice o annulla l'eliminazione

Scenario 3

1. **include Visualizza proteine**
2. L'amministratore inserisce il codice della proteina da eliminare
3. Il sistema verifica se il codice inserito è valido
4. Il sistema verifica se alla proteina è già associato almeno un elemento nella catena della proteina o a qualche mutazione missense
5. Il sistema informa l'amministratore che la proteina non è eliminabile in quanto c'è qualche elemento nella catena della proteina e/o qualche mutazione missense associato ad essa
6. L'amministratore modifica il codice da eliminare o annulla l'eliminazione

Caso d'uso: Inserimento Catena Proteina

Scenario 1

1. **Include Login**
2. L'amministratore inserisce i dati relativi alla catena della proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che la proteina sia esistente
5. Il sistema verifica che il numero di residuo non sia già esistente e che sia valido
6. Il sistema verifica che l'aminoacido sia presente nella banca dati
7. Il sistema verifica che il conservation score sia compreso tra 0 e 1
8. Il sistema registra i dati nella banca dati

Scenario 2

1. **include Login**
2. L'amministratore inserisce i dati relativi alla catena della proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
5. L'amministratore provvede ad inserire i dati obbligatori o annulla l'inserimento

Scenario 3

1. **Include Login**
2. L'amministratore inserisce i dati relativi alla catena della proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che la proteina sia esistente
5. Il sistema informa l'amministratore che la proteina non è esistente
6. L'amministratore provvede ad inserire nuovamente i dati o annulla l'inserimento

Scenario 4

1. **Include Login**
2. L'amministratore inserisce i dati relativi alla catena della proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che la proteina sia esistente
5. Il sistema verifica che il numero di residuo non sia già esistente e che sia valido
6. Il sistema informa l'amministratore che il numero di residuo inserito è già presente per la proteina selezionata e/o se esso è valido
7. L'amministratore provvede a modificare il numero di residuo o annulla l'operazione

Scenario 5

1. **Include Login**
2. L'amministratore inserisce i dati relativi alla catena della proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che la proteina sia esistente
5. Il sistema verifica che il numero di residuo non sia già esistente e che sia valido
6. Il sistema verifica che l'aminoacido sia presente nella banca dati
7. Il sistema informa l'amministratore che l'aminoacido che si vuole inserire non è resente nella banca dati
8. L'amministratore provvede a modificare valore dell'aminoacido o annulla l'inserimento

Scenario 6

1. **Include Login**
2. L'amministratore inserisce i dati relativi alla catena della proteina
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che la proteina sia esistente
5. Il sistema verifica che il numero di residuo non sia già esistente e che sia valido

6. Il sistema verifica che l'aminoacido sia presente nella banca dati
7. Il sistema verifica che il conservation score sia compreso tra 0 e 1
8. Il sistema informa l'amministratore che il valore del conservation score è errato
9. L'amministratore provvede a modificare il valore del conservation score o annulla l'inserimento

Caso d'uso: Visualizzazione Catena proteina

Scenario 1

1. **include Login**
2. L'amministratore richiede l'elenco degli elementi della catena della proteina
3. Il sistema verifica che siano presenti elementi della catena nella banca dati
4. Il sistema fornisce l'elenco degli elementi della catena

Scenario 2

1. **include Login**
2. L'amministratore richiede l'elenco degli elementi della catena della proteina
3. Il sistema verifica che siano presenti elementi della catena nella banca dati
4. Il sistema informa l'Amministratore che non sono presenti elementi nella catena della proteina nella banca dati
5. L'Amministratore annulla l'operazione

Caso d'uso: Modifica catena proteina

Scenario 1

1. **include Visualizzazione Catena proteina**
2. L'amministratore inserisce i dati dell'elemento della catena della proteina da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi all'elemento della catena della proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che il codice della proteina modificato sia presente nella banca dati
7. Il sistema verifica che il numero di residuo modificato non sia già presente nella banca dati
8. Il sistema verifica che il nome dell'aminoacido modificato sia presente nella banca dati
9. Il sistema verifica che il conservation score sia compreso tra 0 e 1
10. Il sistema verifica se ci sono Mutazioni Missense associate all'elemento della catena proteina
11. Il sistema verifica se vi sono Analisi associate all'elemento selezionato e in caso affermativo provvede ad effettuare l'aggiornamento in cascata
12. Il sistema provvede a registrare i nuovi dati

Scenario 2

1. **include Visualizzazione Catena proteina**
2. L'amministratore inserisce i dati dell'elemento della catena della proteina da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. Il sistema informa l'amministratore che i dati inseriti non sono validi
5. L'amministratore provvede ad inserire nuovamente i dati o annulla la modifica

Scenario 3

1. **include Visualizzazione Catena proteina**
2. L'amministratore inserisce i dati dell'elemento della catena della proteina da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi all'elemento della catena della proteina

5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
7. L'amministratore provvede ad inserire i dati obbligatori o annulla la modifica

Scenario 4

1. **include Visualizzazione Catena proteina**
2. L'amministratore inserisce i dati dell'elemento della catena della proteina da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi all'elemento della catena della proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che il codice della proteina modificato sia presente nella banca dati
7. Il sistema informa l'amministratore che la proteina che si sta inserendo non è presente nella banca dati
8. L'amministratore provvede a modificare il nome della proteina o annulla la modifica

Scenario 5

1. **include Visualizzazione Catena proteina**
2. L'amministratore inserisce i dati dell'elemento della catena della proteina da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi all'elemento della catena della proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che il codice della proteina modificato sia presente nella banca dati
7. Il sistema verifica che il numero di residuo modificato non sia già presente nella banca dati
8. Il sistema informa l'amministratore che il numero di residuo che si sta inserendo è già presente nella catena della proteina
9. L'amministratore provvede a modificare il numero di residuo o annulla la modifica

Scenario 6

1. **include Visualizzazione Catena proteina**
2. L'amministratore inserisce i dati dell'elemento della catena della proteina da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi all'elemento della catena della proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che il codice della proteina modificato sia presente nella banca dati
7. Il sistema verifica che il numero di residuo modificato non sia già presente nella banca dati
8. Il sistema verifica che il nome dell'aminoacido modificato sia presente nella banca dati
9. Il sistema informa l'amministratore che l'aminoacido che si tenta di inserire non è presente nella banca dati
10. L'amministratore provvede a modificare l'aminoacido o annulla la modifica

Scenario 7

1. **include Visualizzazione Catena proteina**
2. L'amministratore inserisce i dati dell'elemento della catena della proteina da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi all'elemento della catena della proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che il codice della proteina modificato sia presente nella banca dati
7. Il sistema verifica che il numero di residuo modificato non sia già presente nella banca dati

8. Il sistema verifica che il nome dell'aminoacido modificato sia presente nella banca dati
9. Il sistema verifica che il conservation score sia compreso tra 0 e 1
10. Il sistema informa l'amministratore che il valore di conservation score inserito non è valido
11. L'amministratore modifica il valore del conservation score o annulla la modifica

Scenario 8

1. **include Visualizzazione Catena proteina**
2. L'amministratore inserisce i dati dell'elemento della catena della proteina da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi all'elemento della catena della proteina
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che il codice della proteina modificato sia presente nella banca dati
7. Il sistema verifica che il numero di residuo modificato non sia già presente nella banca dati
8. Il sistema verifica che il nome dell'aminoacido modificato sia presente nella banca dati
9. Il sistema verifica che il conservation score sia compreso tra 0 e 1
10. Il sistema verifica se ci sono Mutazioni Missense associate all'elemento della catena proteina
11. Il sistema informa che non è possibile modificare l'elemento perché ci sono Mutazioni Missense associate
12. L'amministratore provvede a modificare i valori inseriti o annulla la modifica

Caso d'uso: Eliminazione Catena proteina

Scenario 1

1. **include Visualizzazione catena proteina**
2. L'amministratore inserisce i dati relativi all'elemento della catena della proteina da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema verifica l'esistenza dell'elemento nella catena della proteina
5. Il sistema verifica se l'elemento è eliminabile, ovvero se esiste una mutazione missense o qualche Analisi ad esso associate
6. Il sistema elimina l'elemento della catena della proteina

Scenario 2

1. **include Visualizzazione catena proteina**
2. L'amministratore inserisce i dati relativi all'elemento della catena della proteina da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
5. L'amministratore provvede ad inserire i dati obbligatori o annulla l'eliminazione

Scenario 3

1. **include Visualizzazione catena proteina**
2. L'amministratore inserisce i dati relativi all'elemento della catena della proteina da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema verifica l'esistenza dell'elemento nella catena della proteina
5. Il sistema informa l'amministratore che nella catena della proteina non esiste l'elemento inserito
6. L'amministratore provvede ad inserire un altro elemento o annulla l'eliminazione

Scenario 4

1. **include Visualizzazione catena proteina**
2. L'amministratore inserisce i dati relativi all'elemento della catena della proteina da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema verifica l'esistenza dell'elemento nella catena della proteina
5. Il sistema verifica se l'elemento è eliminabile, ovvero se esiste una mutazione missense o qualche Analisi ad esso associate
6. Il sistema informa l'amministratore che l'elemento non è eliminabile perché c'è qualche mutazione missense e/o qualche analisi ad esso associata
7. L'amministratore inserisce un nuovo elemento o annulla l'eliminazione

Caso d'uso: Inserimento mutazione missense

Scenario 1

1. **Include Login**
2. L'amministratore inserisce i dati relativi alla mutazione missense
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che l'elemento della catena della proteina sia presente nella banca dati
5. Il sistema verifica che il nome della mutazione non sia già presente nella banca dati
6. Il sistema ricava, dai valori inseriti, il nome della mutazione
7. Il sistema registra i dati nella banca dati

Scenario 2

1. **Include Login**
2. L'amministratore inserisce i dati relativi alla mutazione missense
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
5. L'amministratore inserisce i dati obbligatori o annulla l'inserimento

Scenario 3

1. **Include Login**
2. L'amministratore inserisce i dati relativi alla mutazione missense
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che l'elemento della catena della proteina sia presente nella banca dati
5. Il sistema informa l'amministratore che l'elemento della catena della proteina inserito non esiste nella banca dati
6. L'amministratore modifica i dati inseriti o annulla l'inserimento

Scenario 4

1. **Include Login**
2. L'amministratore inserisce i dati relativi alla mutazione missense
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica che l'elemento della catena della proteina sia presente nella banca dati
5. Il sistema verifica che il nome della mutazione non sia già presente nella banca dati
6. Il sistema informa l'amministratore che il nome della mutazione è già presente nella banca dati
7. L'amministratore provvede ad inserire una nuova mutazione o annulla l'inserimento

Caso d'uso: Visualizza Mutazione missense

Scenario 1

1. **include Login**
2. L'amministratore richiede l'elenco delle mutazioni missense
3. Il sistema verifica che siano presenti mutazioni missense nella banca dati
4. Il sistema fornisce l'elenco delle mutazioni missense

Scenario 2

1. **include Login**
2. L'amministratore richiede l'elenco delle mutazioni missense
3. Il sistema verifica che siano presenti mutazioni missense nella banca dati
4. Il sistema informa l'Amministratore che non sono presenti mutazioni missense nella banca dati
5. L'Amministratore annulla l'operazione

Caso d'uso: Modifica Mutazione missense

Scenario 1

1. **include Visualizzazione Mutazione missense**
2. L'amministratore inserisce i dati della mutazione da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi alla mutazione
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che l'elemento della catena della proteine modificato sia presente nella banca dati
7. Il sistema verifica, nel caso in cui viene modificato il nome aminoacido(mutante o mutato) e/o il codon (mutante o mutato), che questi siano presenti nella banca dati
8. Il sistema verifica che il nome della mutazione modificato non sia già esistente nella banca dati
9. Il sistema verifica se vi sono analisi in vitro associate alla mutazione che si vuole modificare e in caso affermativo provvede all'aggiornamento in cascata
10. Il sistema provvede a registrare i nuovi dati

Scenario 2

1. **include Visualizzazione Mutazione missense**
2. L'amministratore inserisce i dati della mutazione da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. Il sistema informa l'amministratore che i dati inseriti non sono validi
5. L'amministratore provvede ad inserire nuovamente i dati o annulla l'inserimento

Scenario 3

1. **include Visualizzazione Mutazione missense**
2. L'amministratore inserisce i dati della mutazione da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi alla mutazione
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
7. L'amministratore provvede ad inserire i dati obbligatori o annulla l'inserimento

Scenario 4

1. **include Visualizzazione Mutazione missense**

2. L'amministratore inserisce i dati della mutazione da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi alla mutazione
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che l'elemento della catena della proteine modificato sia presente nella banca dati
7. Il sistema informa l'amministratore che l'elemento della catena della proteina inserito non esiste nella banca dati
8. L'amministratore modifica i dati inseriti o annulla la modifica

Scenario 5

1. **include Visualizzazione Mutazione missense**
2. L'amministratore inserisce i dati della mutazione da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi alla mutazione
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che l'elemento della catena della proteine modificato sia presente nella banca dati
7. Il sistema verifica, nel caso in cui viene modificato il nome aminoacido(mutante o mutato) e/o il codon (mutante o mutato), che questi siano presenti nella banca dati
8. Il sistema informa che il nome aminoacido (mutante o mutato) e/o il codon (mutante o mutato) non è presente nella banca dati
9. L'amministratore provvede a modificare i dati o annulla l'operazione

Scenario 6

1. **include Visualizzazione Mutazione missense**
2. L'amministratore inserisce i dati della mutazione da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi alla mutazione
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che l'elemento della catena della proteine modificato sia presente nella banca dati
7. Il sistema verifica, nel caso in cui viene modificato il nome aminoacido(mutante o mutato) e/o il codon (mutante o mutato), che questi siano presenti nella banca dati
8. Il sistema verifica che il nome della mutazione modificato non sia già esistente nella banca dati
9. Il sistema informa l'amministratore che il nome della mutazione è già presente nella banca dati
10. L'amministratore provvede a modificare i dati della mutazione o annulla la modifica

Caso d'uso: Eliminazione Mutazione missense

Scenario 1

1. **include Visualizzazione Mutazione missense**
2. L'amministratore inserisce i dati relativi alla mutazione da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema verifica l'esistenza della mutazione nella banca dati
5. Il sistema verifica se la mutazione è eliminabile, ovvero se esiste una analisi in vitro ad essa associata
6. Il sistema elimina la mutazione

Scenario 2

1. **include Visualizzazione Mutazione missense**
2. L'amministratore inserisce i dati relativi alla mutazione da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
5. L'amministratore provvede ad inserire i dati obbligatori o annulla l'eliminazione

Scenario 3

1. **include Visualizzazione Mutazione missense**
2. L'amministratore inserisce i dati relativi alla mutazione da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema verifica l'esistenza della mutazione nella banca dati
5. Il sistema informa l'amministratore che la mutazione che si vuole eliminare non è presente nella banca dati
6. L'amministratore fornisce un'altra mutazione da eliminare o annulla l'eliminazione

Scenario 4

1. **include Visualizzazione Mutazione missense**
2. L'amministratore inserisce i dati relativi alla mutazione da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema verifica l'esistenza della mutazione nella banca dati
5. Il sistema verifica se la mutazione è eliminabile, ovvero se esiste una analisi in vitro ad essa associata
6. Il sistema informa l'amministratore che la mutazione non è eliminabile, in quanto c'è almeno una analisi in vitro ad essa associata
7. L'amministratore provvede a modificare la mutazione da eliminare o annulla l'eliminazione

➢ *NOTA* I casi d'uso seguenti: Inserimento analisi, Visualizzazione analisi, Modifica analisi, Eliminazione analisi, sono identici per tutte le analisi sugli elementi della proteina elencate:

i. *DSSP*
ii. *NACCESS Monomer*
iii. *NACCESS Dimer*
iv. *H-BOND con Ligando*
v. *Interazioni con Ligando*
vi. *H-BOND con Sidechain*

Pertanto verranno scritte per una generica analisi ma saranno utilizzati per tutte le analisi.

Caso d'uso: Inserimento Analisi

Scenario 1

1. **Include Login**
2. L'amministratore inserisce i dati relativi all'analisi
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica la validità dei dati inseriti
5. Il sistema verifica l'esistenza dell'elemento della catena della proteina su cui è effettuata l'analisi
6. Il sistema verifica che non risultino già memorizzate analisi dello stesso tipo per l'elemento della catena selezionato
7. Il sistema registra i dati nella banca dati

Scenario 2

1. **Include Login**
2. L'amministratore inserisce i dati relativi all'analisi
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
5. L'amministratore inserisce i dati obbligatori o annulla l'inserimento

Scenario 3

1. **Include Login**
2. L'amministratore inserisce i dati relativi all'analisi
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica la validità dei dati inseriti
5. Il sistema informa l'amministratore che i dati inseriti non sono validi
6. L'amministratore provvede a modificare i dati o annulla l'inserimento

Scenario 4

1. **Include Login**
2. L'amministratore inserisce i dati relativi all'analisi
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica la validità dei dati inseriti
5. Il sistema verifica l'esistenza dell'elemento della catena della proteina su cui è effettuata l'analisi
6. Il sistema informa l'amministratore che l'elemento della catena della proteina non esiste nella banca dati
7. L'amministratore provvede a modificare i dati o annulla l'inserimento

Scenario 5

1. **Include Login**
2. L'amministratore inserisce i dati relativi all'analisi
3. Il sistema verifica l'inserimento dei dati obbligatori
4. Il sistema verifica la validità dei dati inseriti
5. Il sistema verifica l'esistenza dell'elemento della catena della proteina su cui è effettuata l'analisi
6. Il sistema verifica che non risultino già memorizzate analisi dello stesso tipo per l'elemento della catena selezionato
7. Il sistema informa l'amministratore che per l'elemento della catena selezionato sono già presenti analisi dello stesso tipo
8. L'amministratore provvede a modificare i dati o annulla l'inserimento

Caso d'uso: Visualizzazione Analisi

Scenario 1

1. **include Login**
2. L'amministratore richiede l'elenco delle analisi effettuate
3. Il sistema verifica che siano presenti analisi
4. Il sistema fornisce l'elenco delle analisi effettuate

Scenario 2

1. **include Login**
2. L'amministratore richiede l'elenco delle analisi effettuate
3. Il sistema verifica che siano presenti analisi
4. Il sistema informa l'Amministratore che non sono presenti analisi nella banca dati
5. L'Amministratore annulla l'operazione

Caso d'uso: Modifica Analisi

Scenario 1

1. **include Visualizzazione Analisi**
2. L'amministratore inserisce i dati dell'analisi da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi alla analisi
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che i dati modificati sono validi
7. Il sistema verifica l'esistenza nella banca dati dell'elemento della catena della proteina modificato
8. Il sistema provvede a registrare i nuovi dati

Scenario 2

1. **include Visualizzazione Analisi**
2. L'amministratore inserisce i dati dell'analisi da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. Il sistema informa l'amministratore che i dati inseriti non sono validi
5. L'amministratore provvede ad inserire nuovamente i dati o annulla l'inserimento

Scenario 3

1. **include Visualizzazione Analisi**
2. L'amministratore inserisce i dati dell'analisi da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi alla analisi
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
7. L'amministratore provvede ad inserire i dati obbligatori o annulla l'inserimento

Scenario 4

1. **include Visualizzazione Analisi**
2. L'amministratore inserisce i dati dell'analisi da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi alla analisi
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che i dati modificati sono validi
7. Il sistema informa l'amministratore che i dati modificati non sono validi
8. L'amministratore modifica i dati inseriti o annulla la modifica

Scenario 5

1. **include Visualizzazione Analisi**
2. L'amministratore inserisce i dati dell'analisi da modificare
3. Il sistema verifica se i dati inseriti sono validi
4. L'amministratore modifica i dati relativi alla analisi
5. Il sistema verifica che tutti i dati obbligatori siano stati inseriti
6. Il sistema verifica che i dati modificati sono validi
7. Il sistema verifica l'esistenza nella banca dati dell'elemento della catena della proteina modificato
8. Il sistema informa l'amministratore che l'elemento della catena della proteina modificato non è presente nella banca dati
9. L'amministratore modifica i dati inseriti o annulla la modifica

Caso d'uso: Eliminazione Analisi

Scenario 1

1. **include Visualizzazione Analisi**
2. L'amministratore inserisce i dati relativi all'analisi da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema verifica l'esistenza dell'analisi nella banca dati
5. Il sistema elimina l'analisi mutazione

Scenario 2

1. **include Visualizzazione Analisi mutazione**
2. L'amministratore inserisce i dati relativi all'analisi da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
5. L'amministratore provvede ad inserire i dati obbligatori o annulla l'eliminazione

Scenario 3

1. **include Visualizzazione Analisi mutazione**
2. L'amministratore inserisce i dati relativi all'analisi sulla mutazione da eliminare
3. Il sistema verifica che i dati obbligatori siano stati inseriti
4. Il sistema verifica l'esistenza dell'analisi nella banca dati
5. Il sistema informa l'amministratore che l'analisi che si vuole eliminare non è presente nella banca dati
6. L'amministratore fornisce un'altra analisi da eliminare o annulla l'eliminazione

Caso d'uso: Logout

Scenario 1

1. **include Login**
2. L'amministratore procede al Logout, ovvero all'uscita dalle pagine di amministrazione
3. Il sistema effettua il Logout dell'amministratore

Caso d'uso: Submit nuova mutazione

Scenario 1

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema verifica che Aminoacido Mutante e Mutato non coincidano

5. Il sistema verifica che Codon Mutante e Mutato non coincidano
6. Il sistema verifica l'esistenza della proteina nella banca dati
7. Il sistema verifica che il residuo e l'Aminoacido Mutante corrispondano ad un elemento della catena della proteina
8. **Include Invio e-mail all'Amministratore**

Scenario 2

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema informa l'utente che i dati obbligatori non sono stati inseriti
4. L'utente inserisce i dati obbligatori o annulla la sottomissione

Scenario 3

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema informa l'utente che l'indirizzo E-mail inserito non è corretto
5. L'utente modifica i dati o annulla la sottomissione

Scenario 4

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema verifica che Aminoacido Mutante e Mutato non coincidano
5. Il sistema informa l'utente che gli Aminoacidi inseriti coincidono
6. L'utente modifica i dati o annulla la sottomissione

Scenario 5

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema verifica che Aminoacido Mutante e Mutato non coincidano
5. Il sistema verifica che Codon Mutante e Mutato non coincidano
6. Il sistema informa l'utente che i Codon inseriti coincidono
7. L'utente modifica i dati o annulla la sottomissione

Scenario 6

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema verifica che Aminoacido Mutante e Mutato non coincidano
5. Il sistema verifica che Codon Mutante e Mutato non coincidano
6. Il sistema verifica l'esistenza della proteina nella banca dati
7. Il sistema informa l'utente che la proteina inserita non esiste nella banca dati
8. L'utente modifica i dati o annulla la sottomissione

Scenario 7

1. L'utente inserisce i dati relativi alla nuova mutazione che vuole sottomettere all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica la correttezza dell'indirizzo E-mail del sottoponente
4. Il sistema verifica che Aminoacido Mutante e Mutato non coincidano
5. Il sistema verifica che Codon Mutante e Mutato non coincidano
6. Il sistema verifica l'esistenza della proteina nella banca dati
7. Il sistema verifica che il residuo e l'Aminoacido Mutante corrispondano ad un elemento della catena della proteina
8. Il sistema informa l'utente che il residuo e l'Aminoacido Mutante non corrispondono a nessun elemento della catena della proteina
9. L'utente modifica l'e-mail o annulla la sottomissione

Caso d'uso: Invia e-mail all'amministratore

Scenario 1

1. L'utente inserisce i dati relativi alle informazioni che vuole inviare all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica che l'indirizzo e-mail inserito sia valido
4. Il sistema preleva l'indirizzo e-mail dell'amministratore
5. Il sistema invia l'e-mail all'indirizzo dell'amministratore

Scenario 2

1. L'utente inserisce i dati relativi alle informazioni che vuole inviare all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema informa l'utente che i dati obbligatori non sono stati inseriti
4. L'utente provvede ad inserire i dati obbligatori o annulla l'invio dell'e-mail

Scenario 3

1. L'utente inserisce i dati relativi alle informazioni che vuole inviare all'amministratore
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica che l'indirizzo e-mail inserito sia valido
4. Il sistema informa l'utente che l'indirizzo e-mail inserito non è valido
5. L'utente modifica l'indirizzo e-mail o ne annulla l'invio

Caso d'uso: Visualizza bibliografia

Scenario 1

1. L'utente richiede la visualizzazione delle informazioni relative ai riferimenti bibliografici utilizzati per lo studio della proteina
2. Il sistema visualizza le informazioni richieste

Caso d'uso: Visualizza link utili

Scenario 1

1. L'utente richiede la visualizzazione delle informazioni relative ai link utili sulla proteina
2. Il sistema visualizza le informazioni richieste

Caso d'uso: Consultazione informazioni sull'intera proteina

Scenario 1

1. L'utente inserisce i dati relativi alla proteina che vuole visualizzare
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica che la proteina inserita sia esistente

4. Il sistema verifica che ci siano elementi nella catena della proteina che soddisfano i parametri immessi
5. Il sistema visualizza le informazioni richieste

Scenario 2
1. L'utente inserisce i dati relativi alla proteina che vuole visualizzare
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
4. L'utente provvede ad inserire i dati obbligatori o annulla l'operazione

Scenario 3
1. L'utente inserisce i dati relativi alla proteina che vuole visualizzare
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica che la proteina inserita sia esistente
4. Il sistema informa l'amministratore che la proteina inserita non esiste
5. L'utente sostituisce il nome della proteina o annulla l'operazione

Scenario 4
1. L'utente inserisce i dati relativi alla proteina che vuole visualizzare
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica che la proteina inserita sia esistente
4. Il sistema verifica che ci siano elementi nella catena della proteina che soddisfano i parametri immessi
5. Il sistema informa l'amministratore che non esistono elementi nella catena della proteina che soddisfano i parametri inseriti
6. L'utente modifica i parametri di ricerca o annulla l'operazione

Caso d'uso: Consultazione informazioni sulle mutazioni

Scenario 1
1. L'utente inserisce i dati relativi alle mutazioni che vuole visualizzare
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica che la proteina inserita sia esistente
4. Il sistema verifica che ci siano Mutazioni Missense che soddisfano i parametri immessi
5. Il sistema visualizza le informazioni richieste

Scenario 2
1. L'utente inserisce i dati relativi alle mutazioni che vuole visualizzare
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema informa l'amministratore che i dati obbligatori non sono stati inseriti
4. L'utente provvede ad inserire i dati obbligatori o annulla l'operazione

Scenario 3
1. L'utente inserisce i dati relativi alle mutazioni che vuole visualizzare
2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica che la proteina inserita sia esistente
4. Il sistema informa l'amministratore che la proteina inserita non esiste
5. L'utente sostituisce il nome della proteina o annulla l'operazione

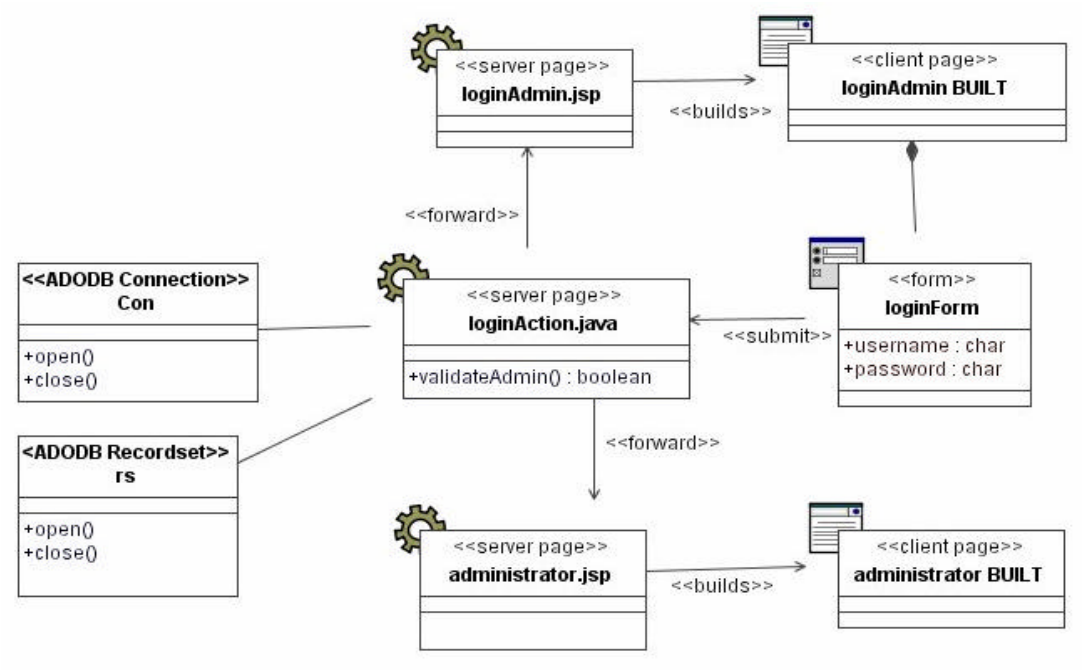
Scenario 4
1. L'utente inserisce i dati relativi alle mutazioni che vuole visualizzare

2. Il sistema verifica che i dati obbligatori siano stati inseriti
3. Il sistema verifica che la proteina inserita sia esistente
4. Il sistema verifica che ci siano Mutazioni Missense che soddisfano i parametri immessi
5. Il sistema informa l'amministratore che non esistono Mutazioni Missense che soddisfano i parametri inseriti
6. L'utente modifica i parametri di ricerca o annulla l'operazione

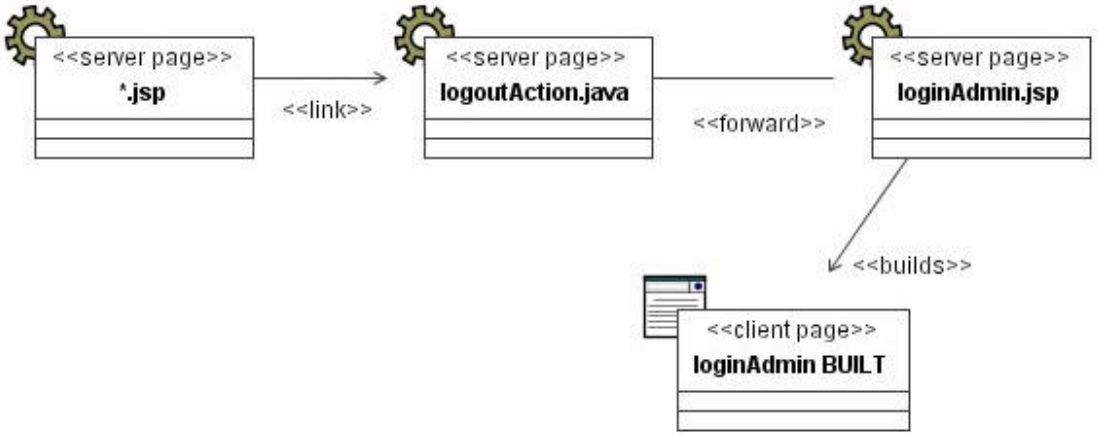
B.3 Class Diagrams

❖ Modulo Amministrazione

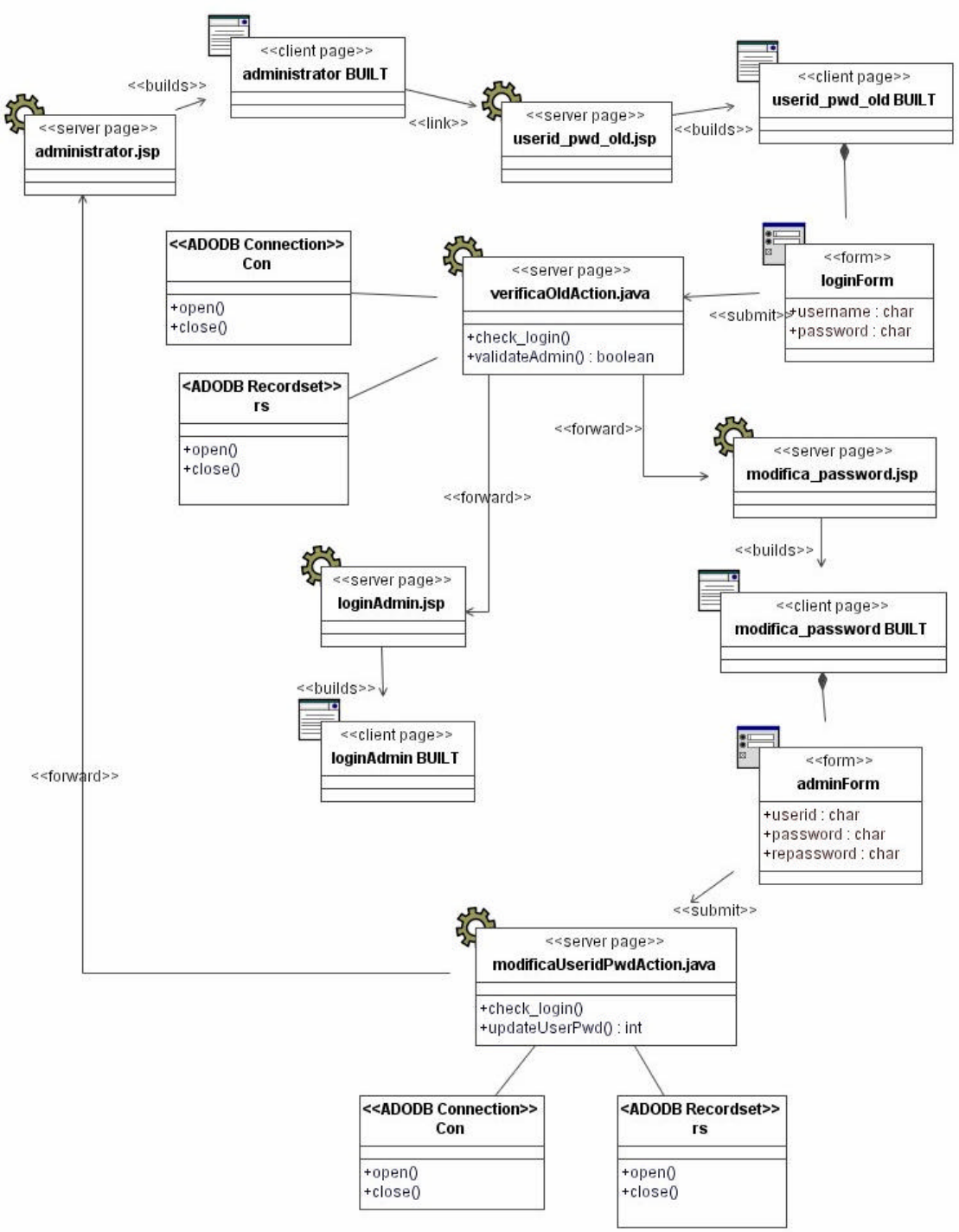
Caso d'uso: Login



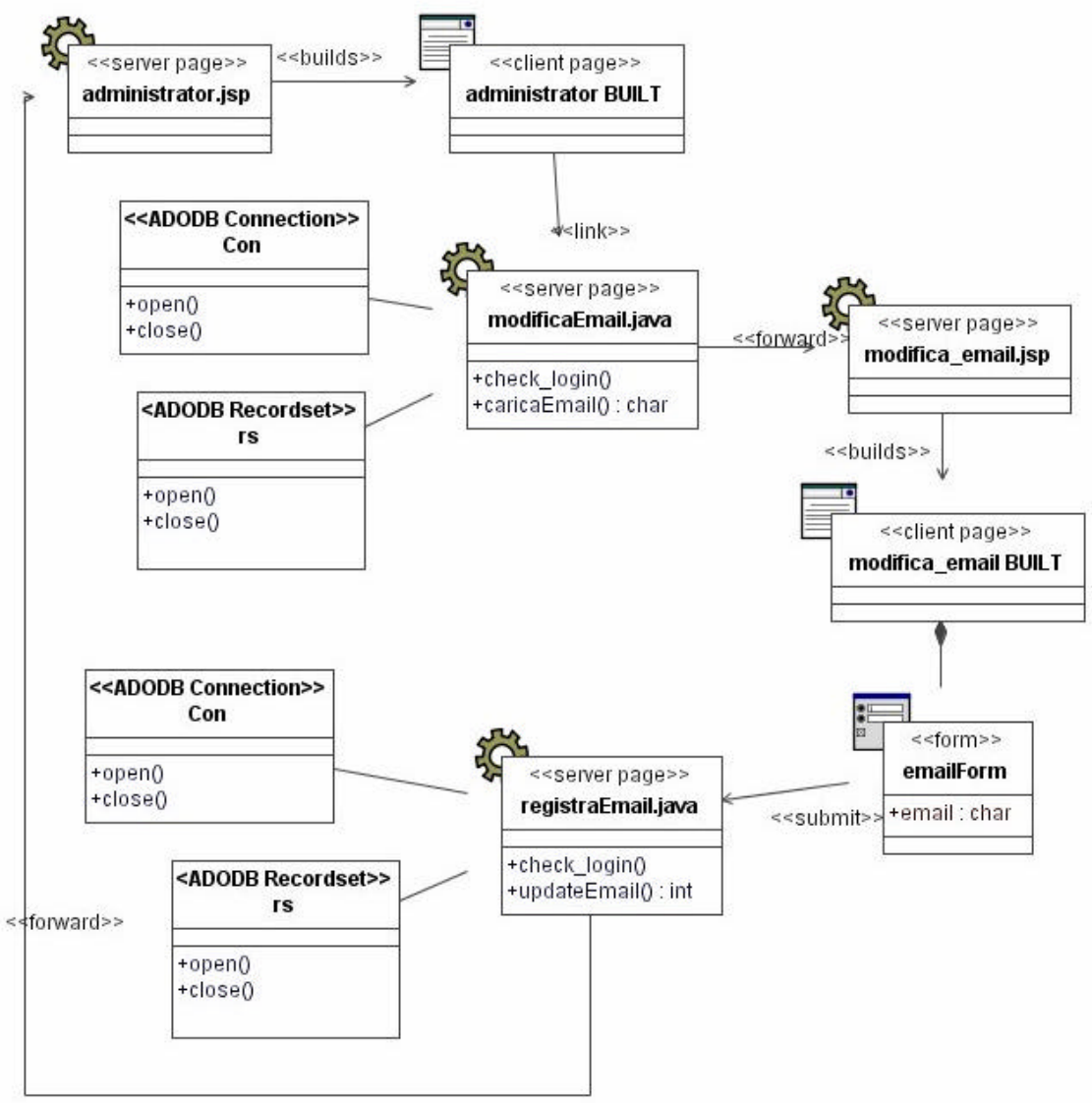
Caso d'uso: Logout



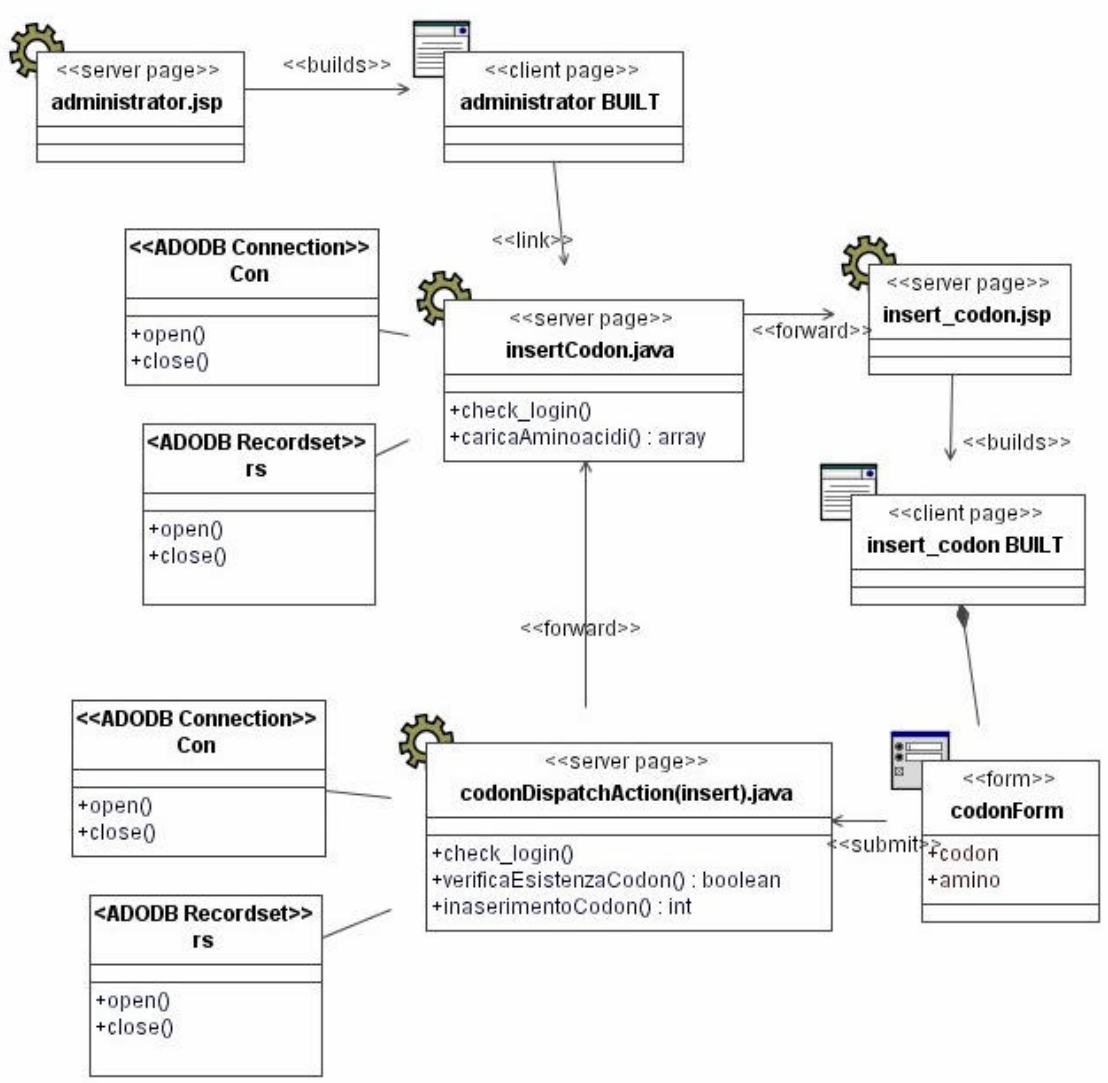
Caso d'uso: modifica username/password di amministrazione



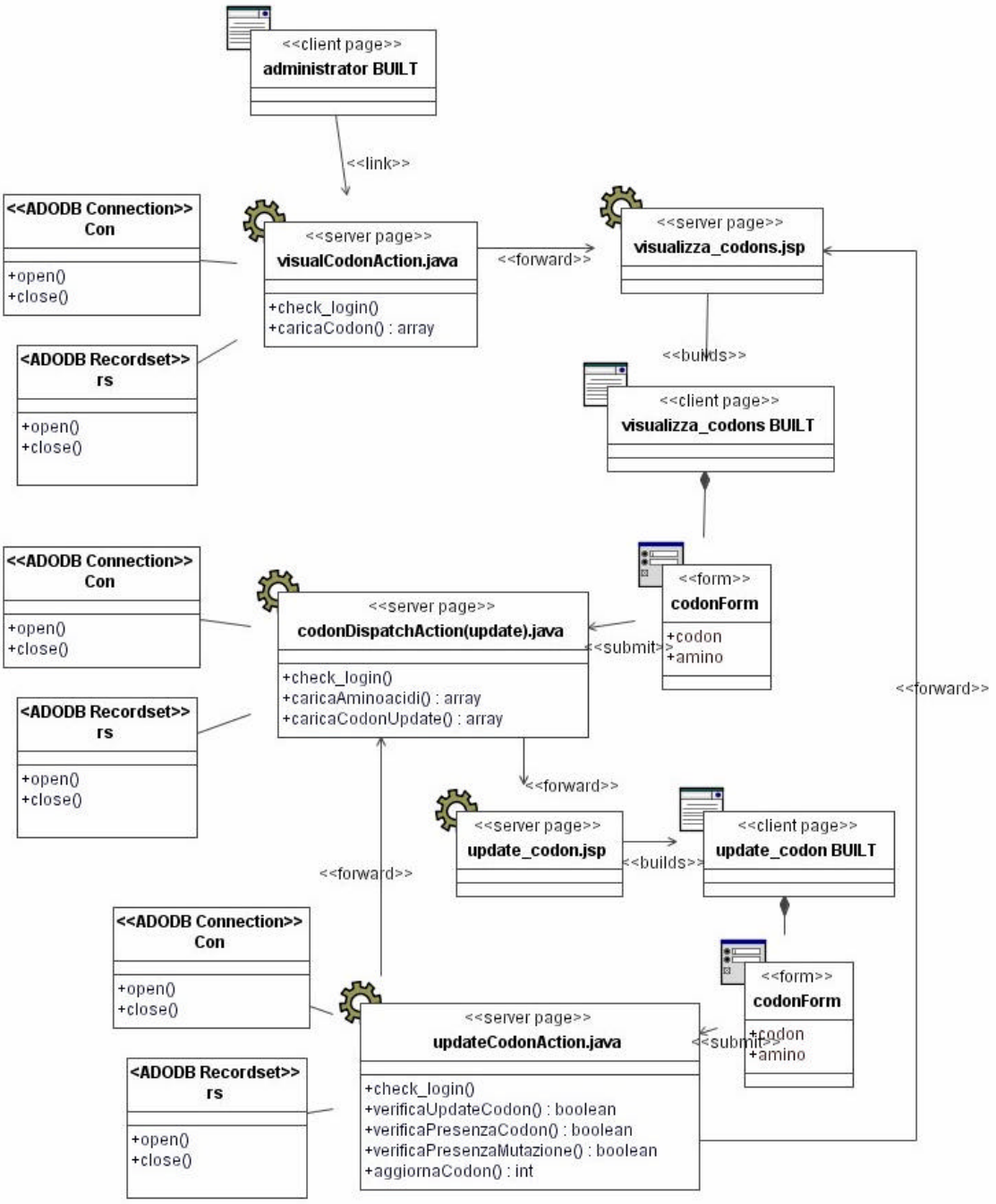
Caso d'uso: modifica email amministratore



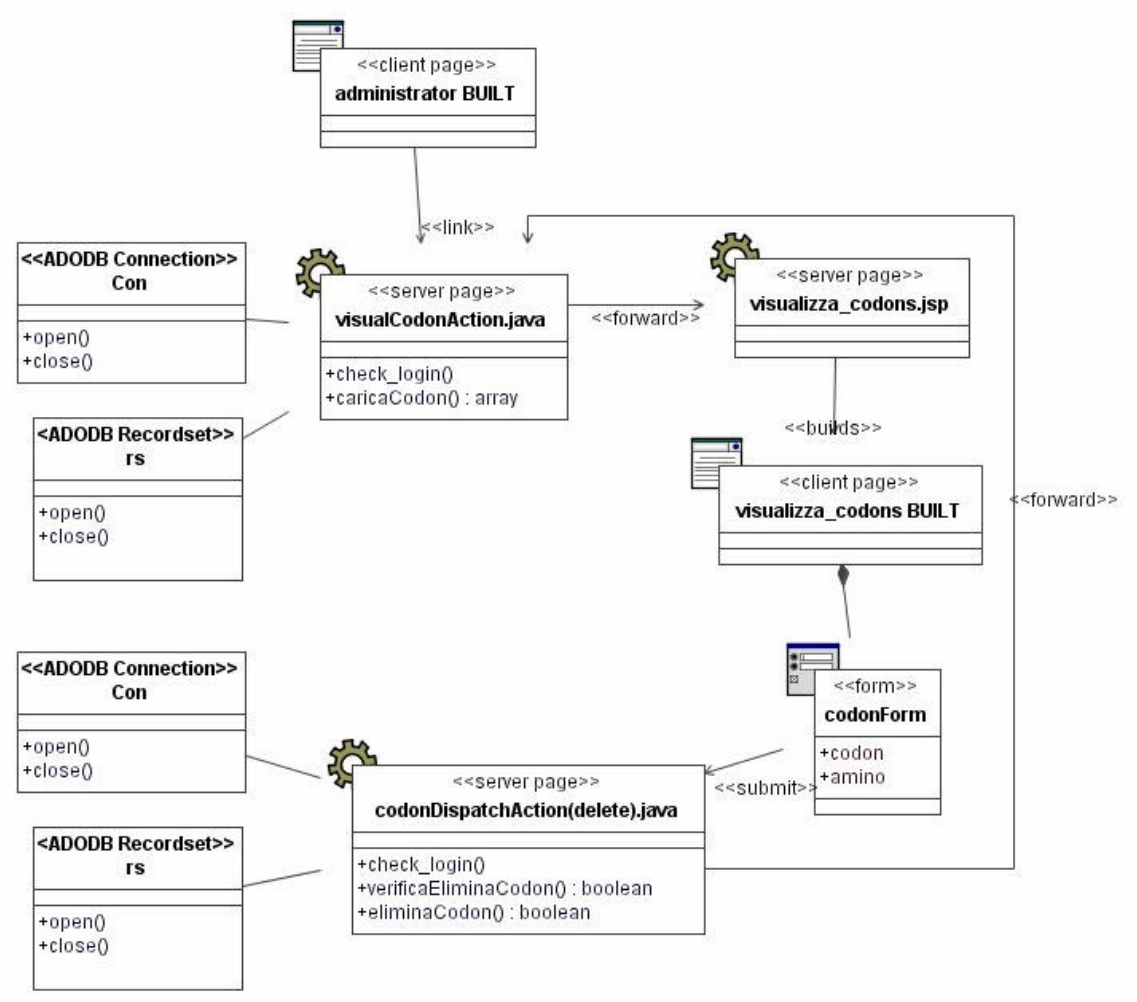
Caso d'uso: Inserimento codon



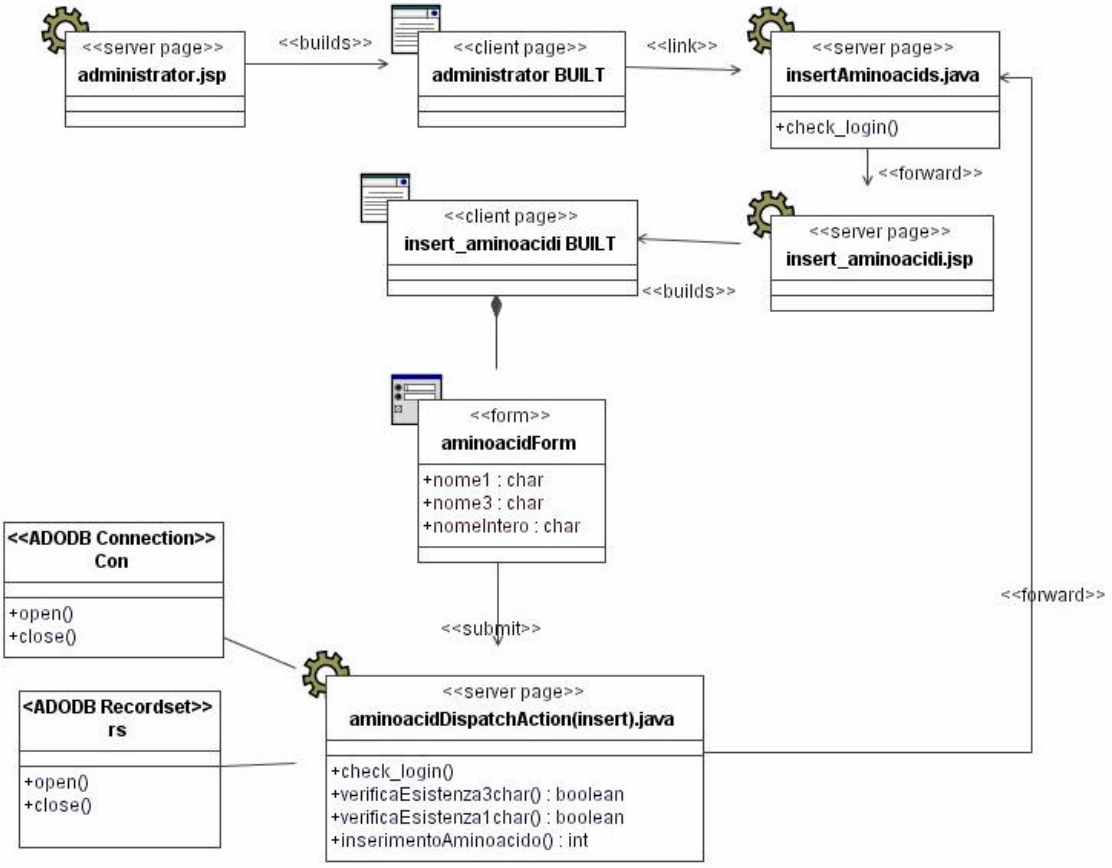
Caso d'uso: Visualizzazione codon e Modifica codon



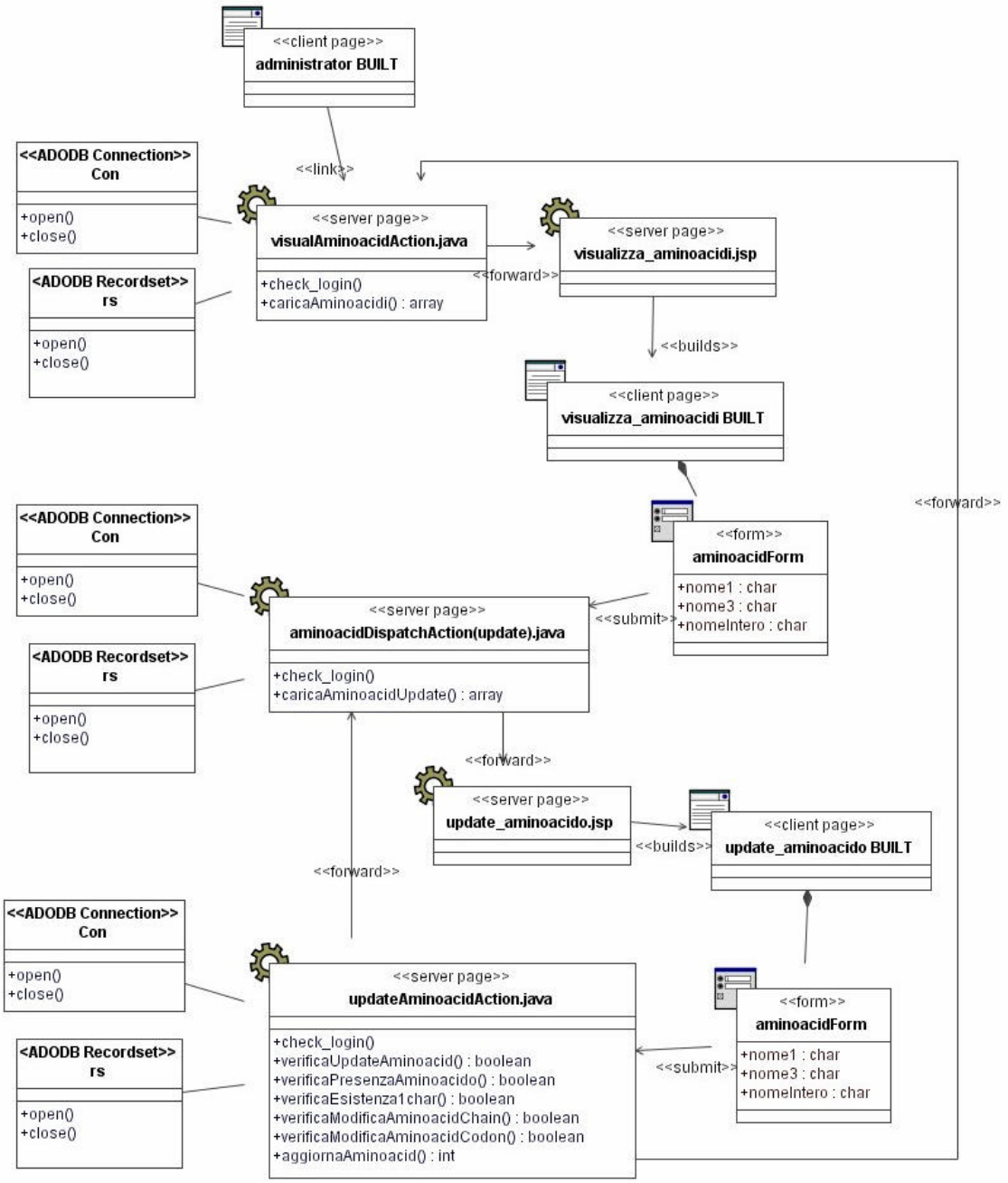
Caso d'uso: Visualizzazione codon ed Eliminazione codon



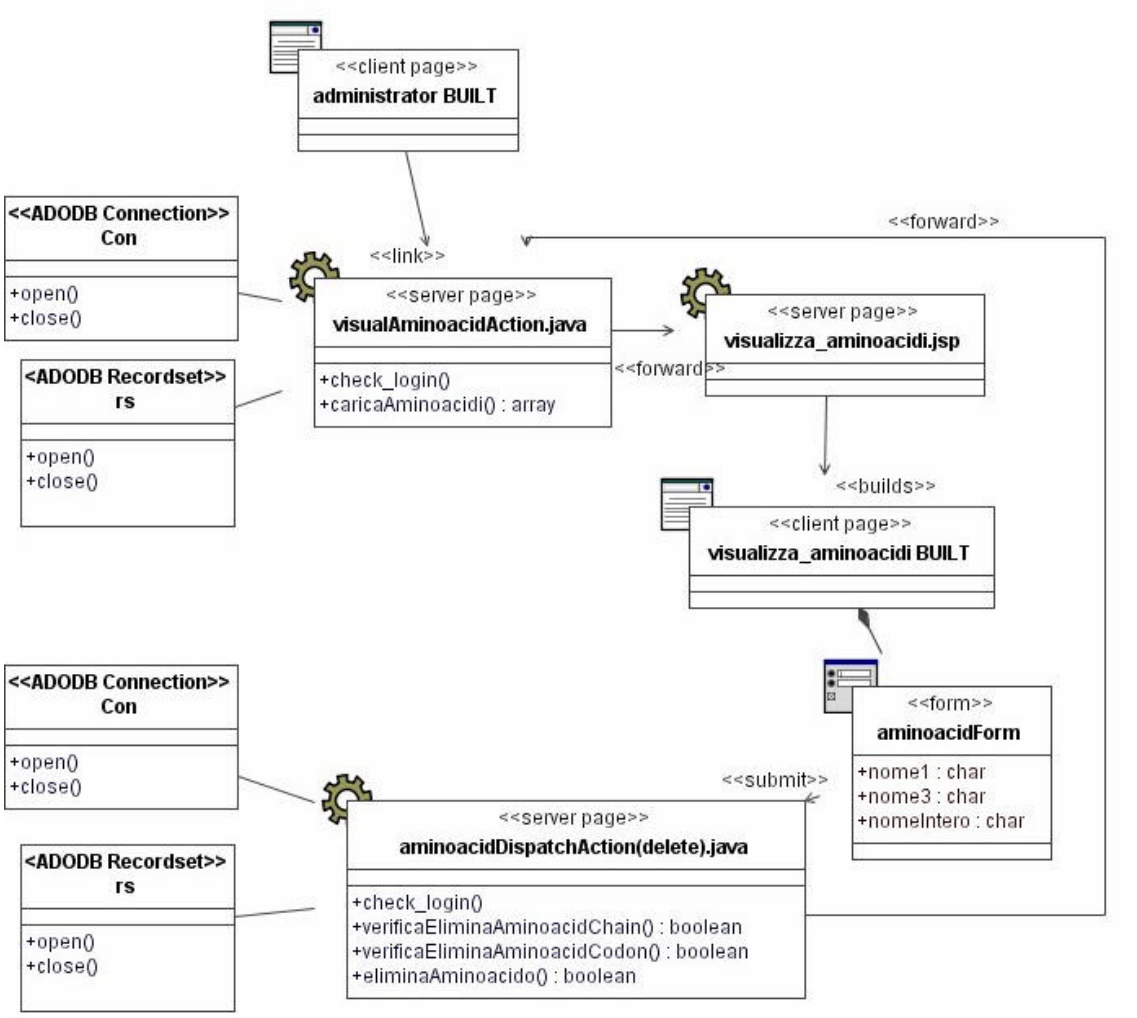
Caso d'uso: Inserimento aminoacido



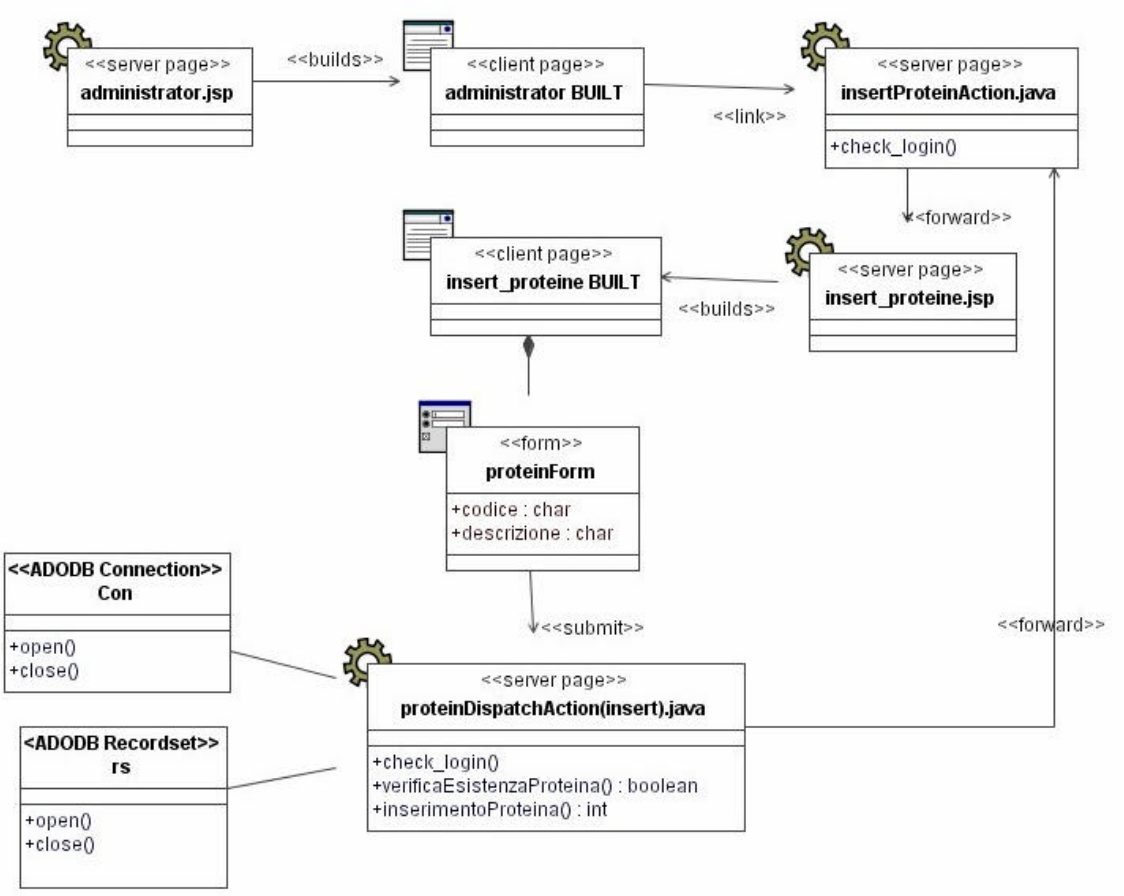
Caso d'uso: Visualizzazione aminoacido e Modifica aminoacido



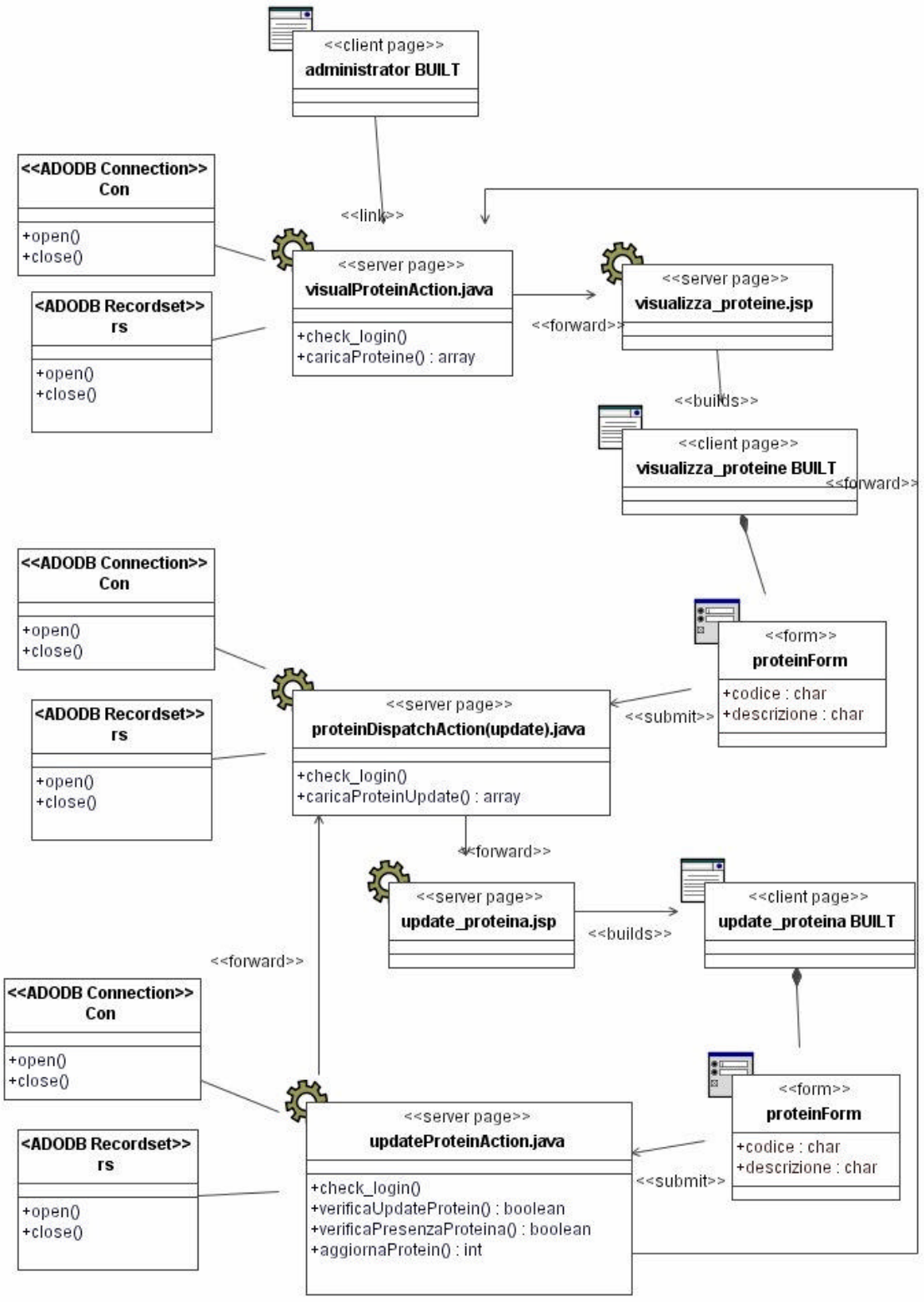
Caso d'uso: Visualizzazione aminoacido ed Eliminazione aminoacido



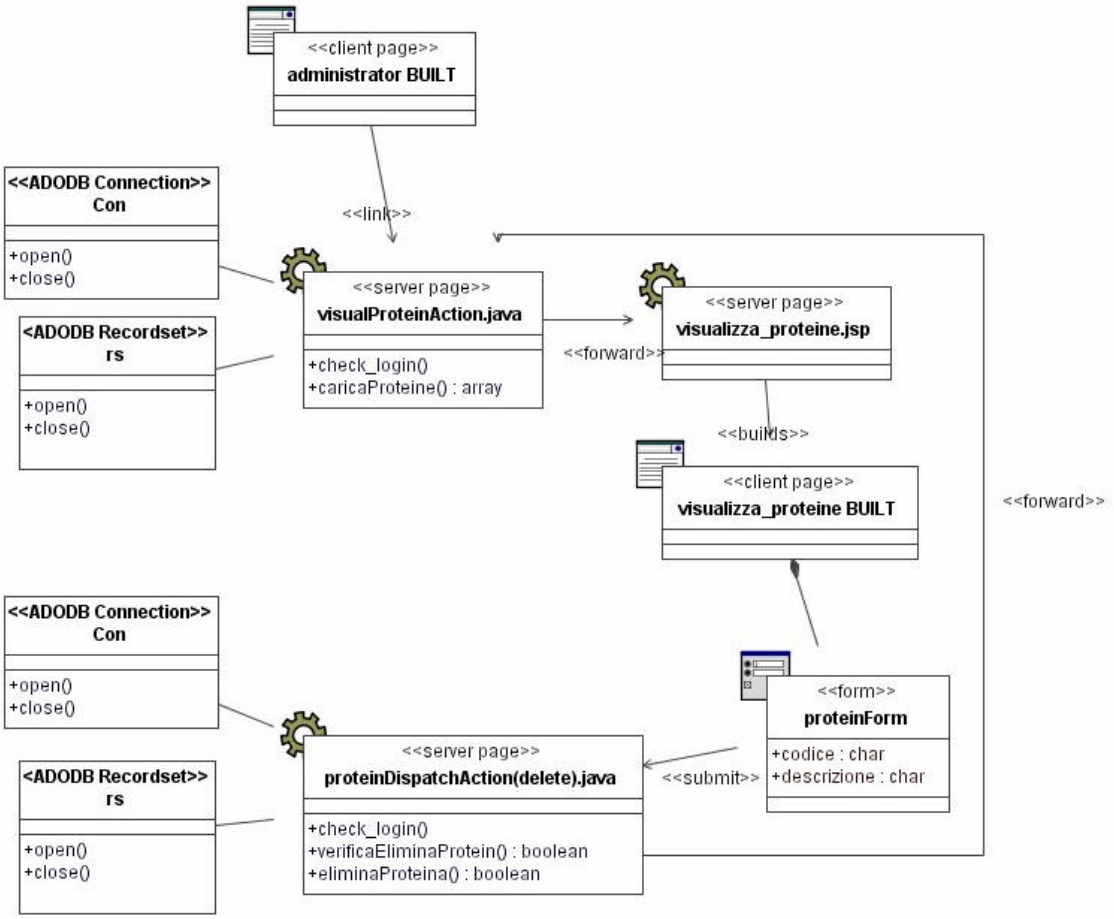
Caso d'uso: Inserimento Proteina



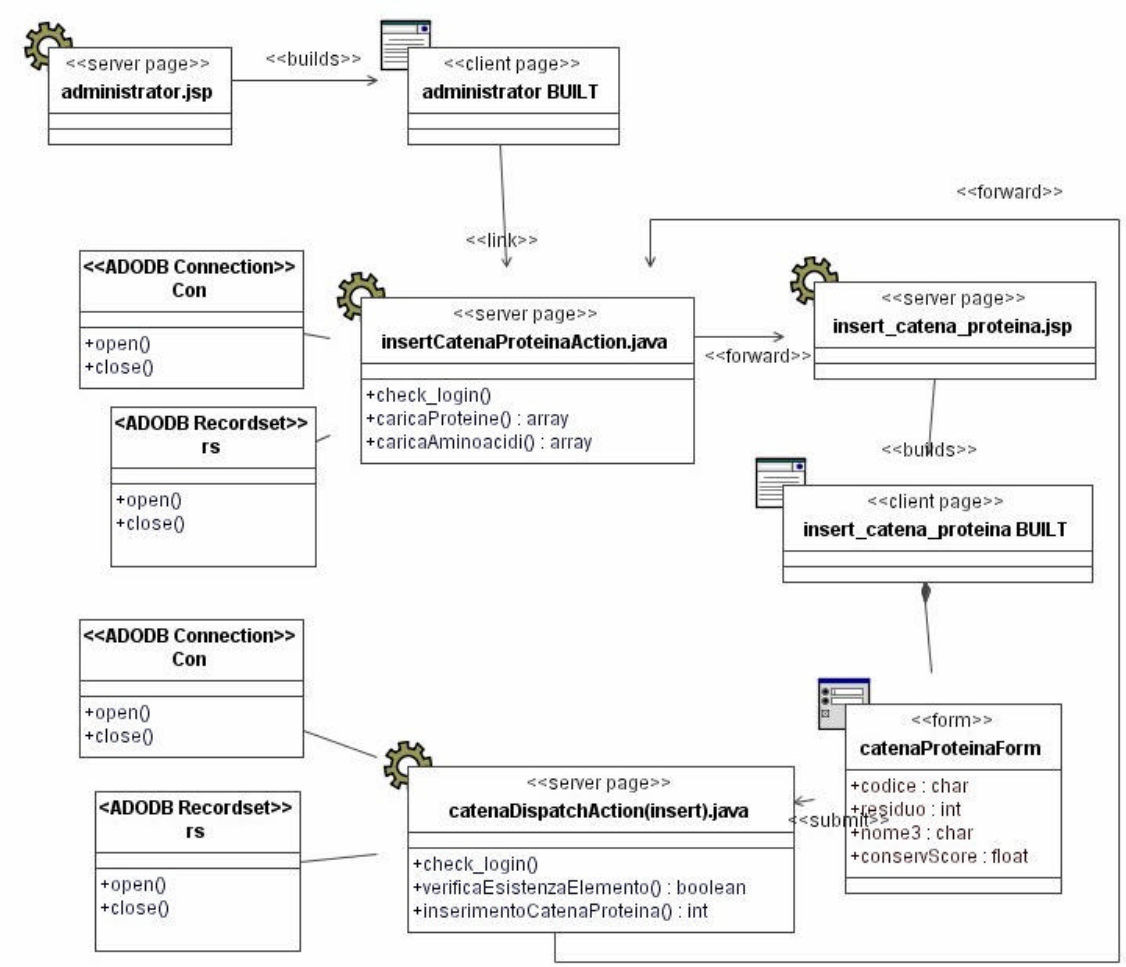
Caso d'uso: Visualizza proteine e Modifica proteine



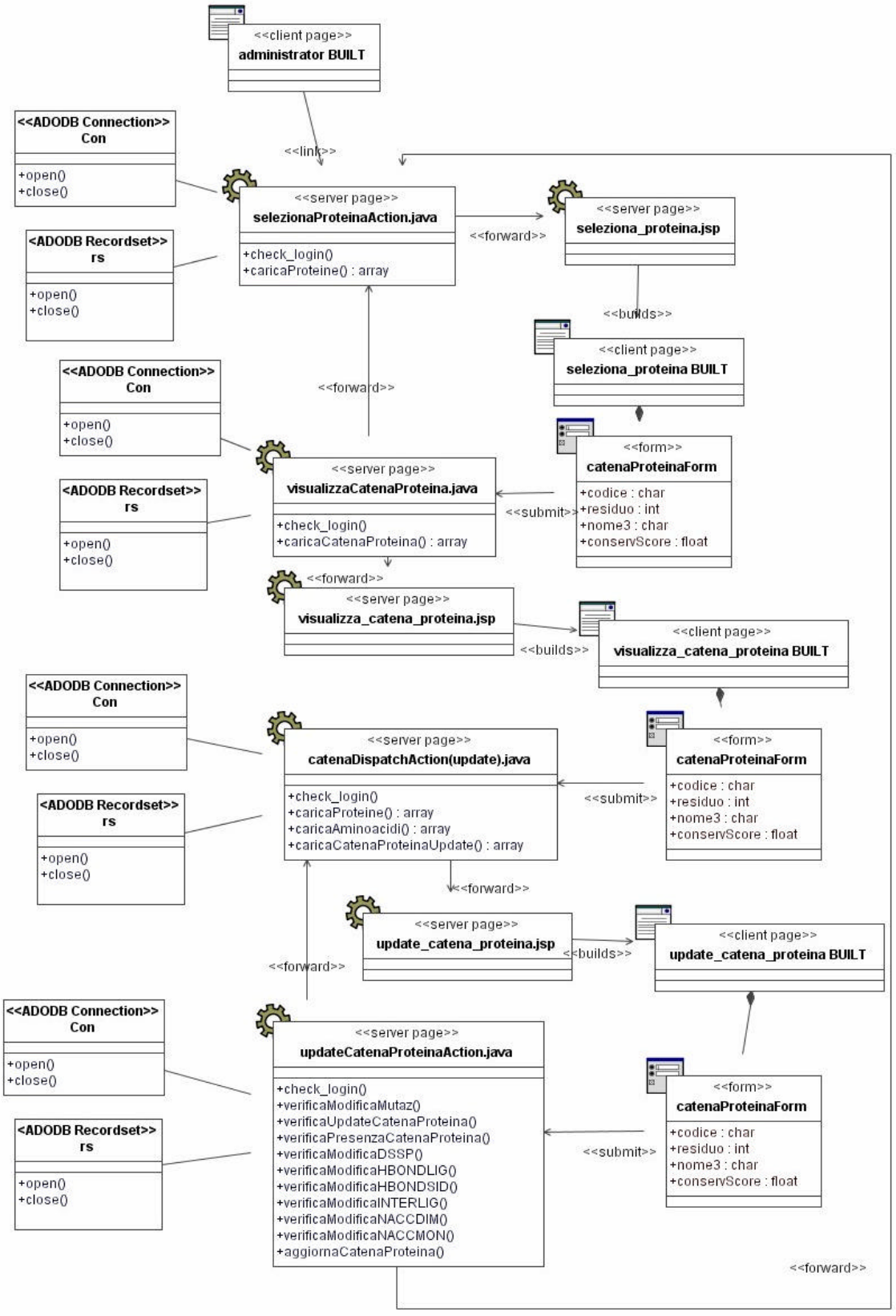
Caso d'uso: Visualizza proteine ed Elimina proteine



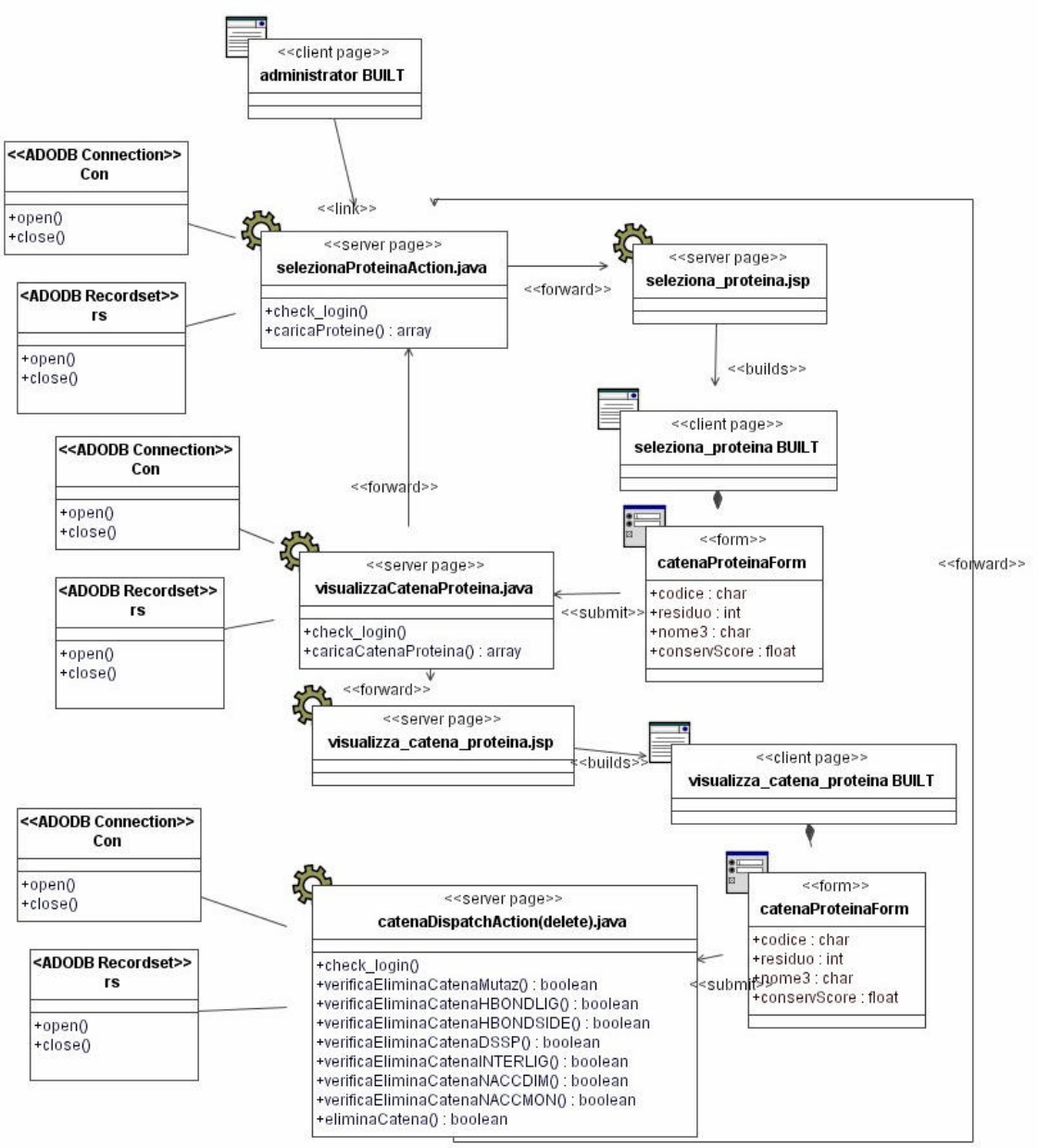
Caso d'uso: Inserimento Catena Proteina



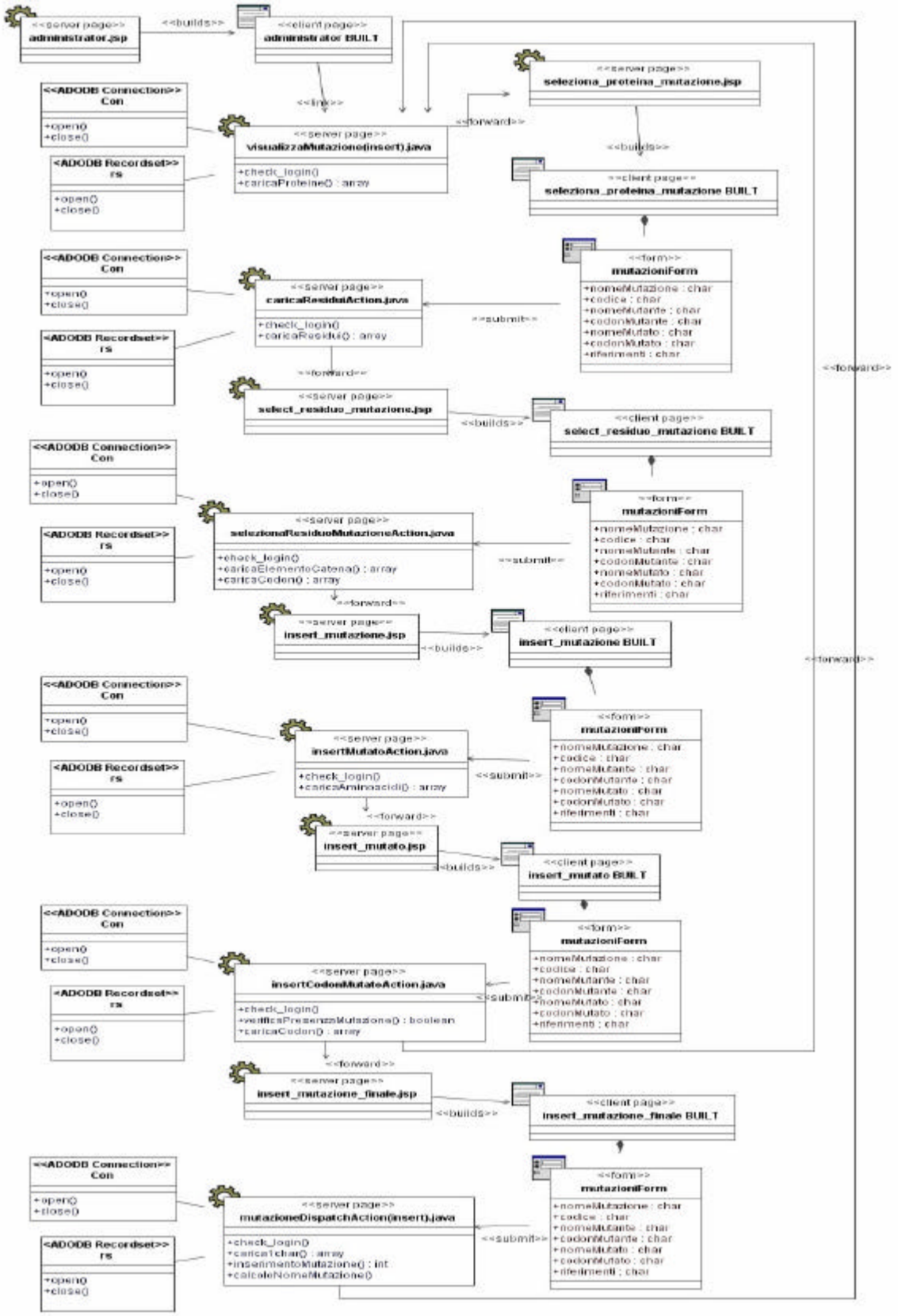
Caso d'uso: Visualizzazione Catena proteina e Modifica Catena proteina



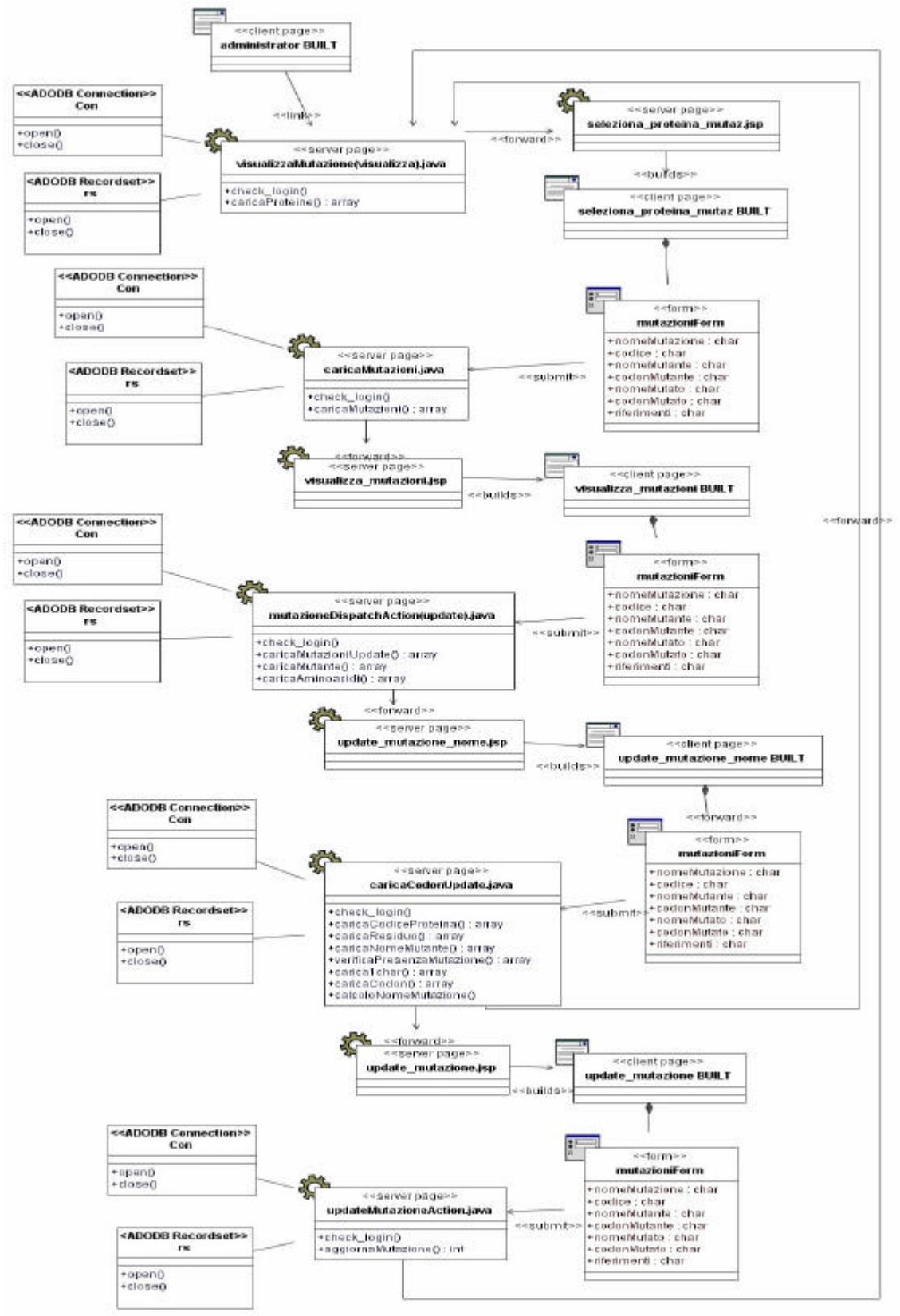
Caso d'uso: Visualizzazione Catena proteina ed Eliminazione Catena proteina



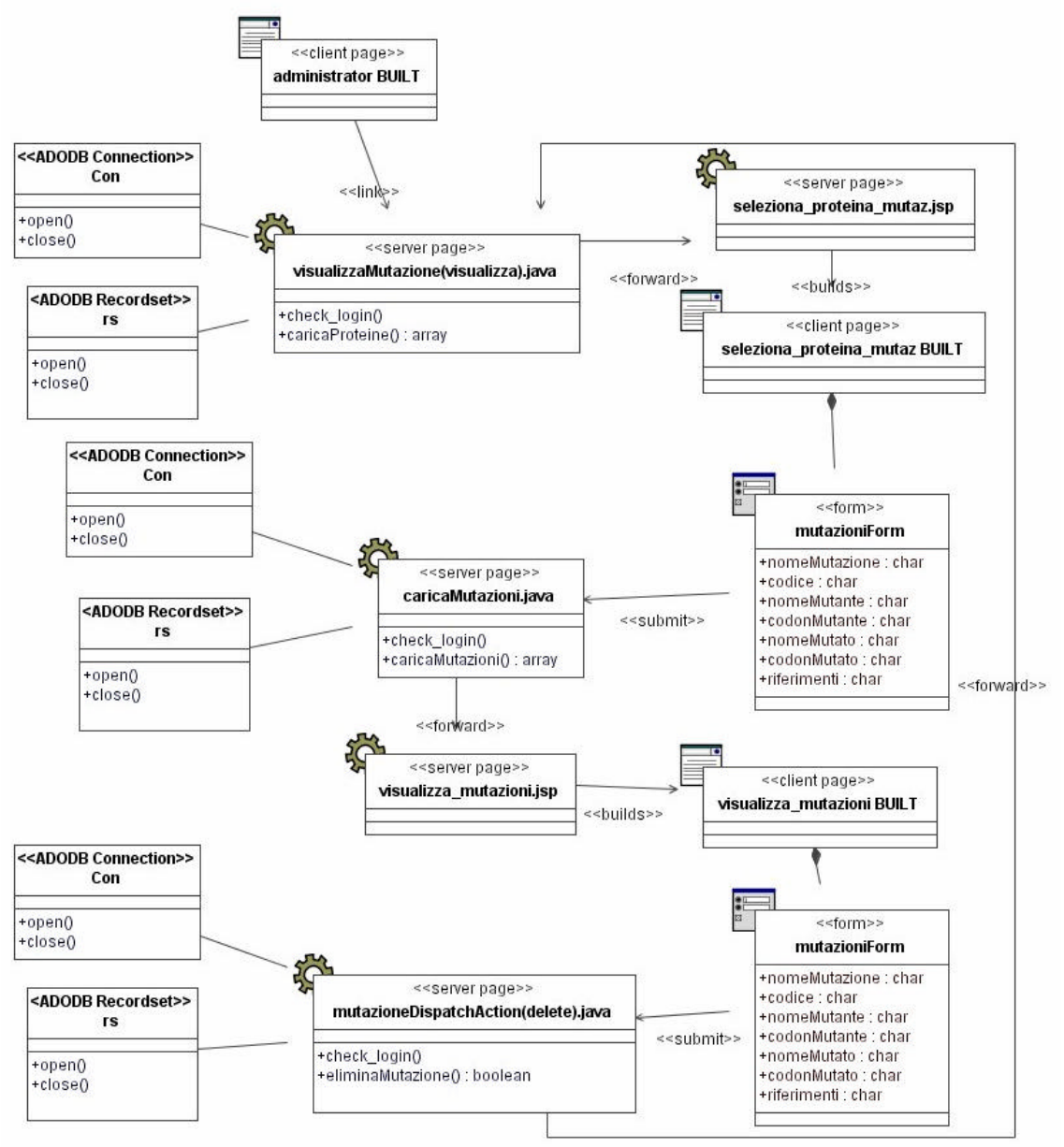
Caso d'uso: Inserimento mutazione missense



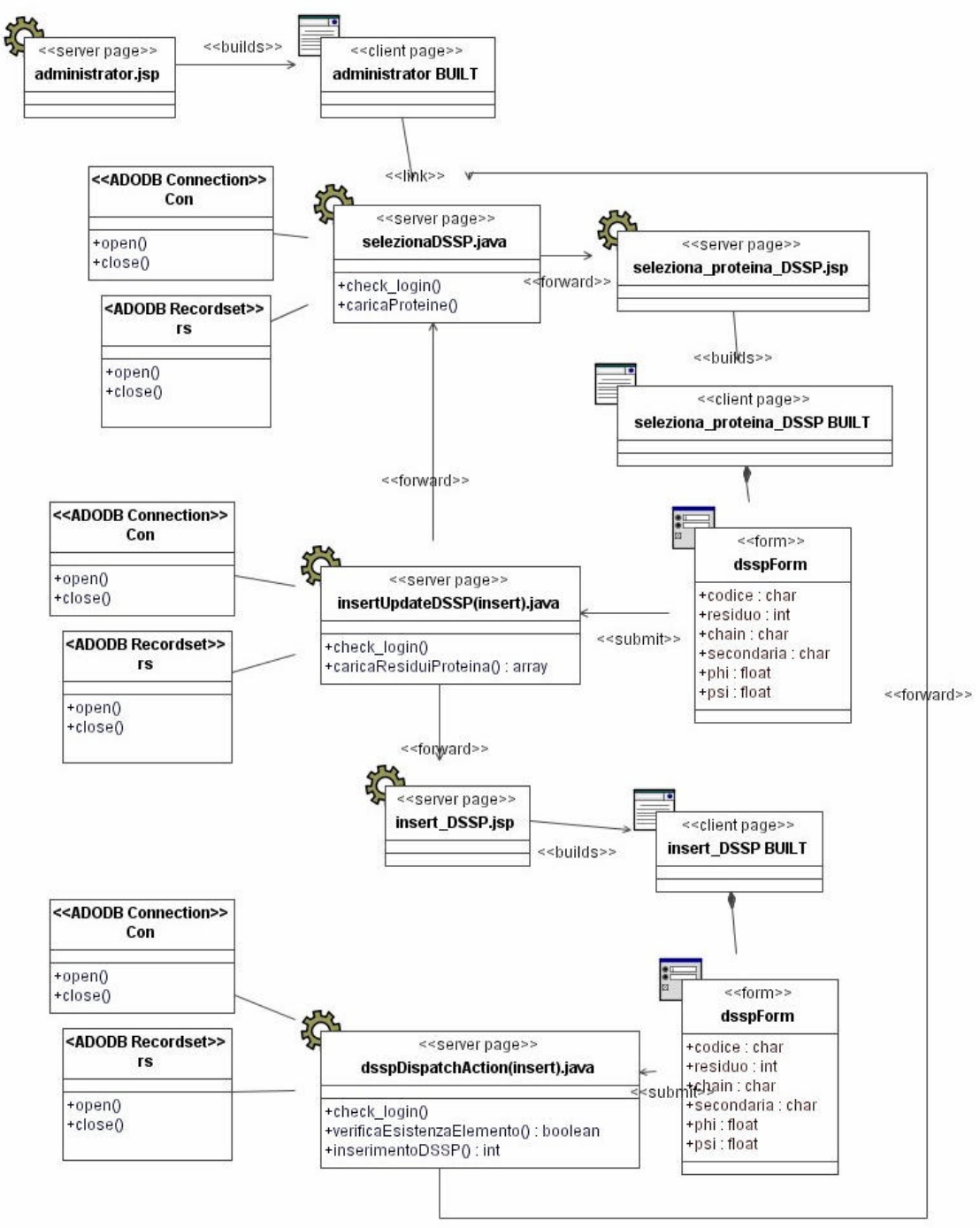
Caso d'uso: Visualizza Mutazione missense e Modifica Mutazione missense



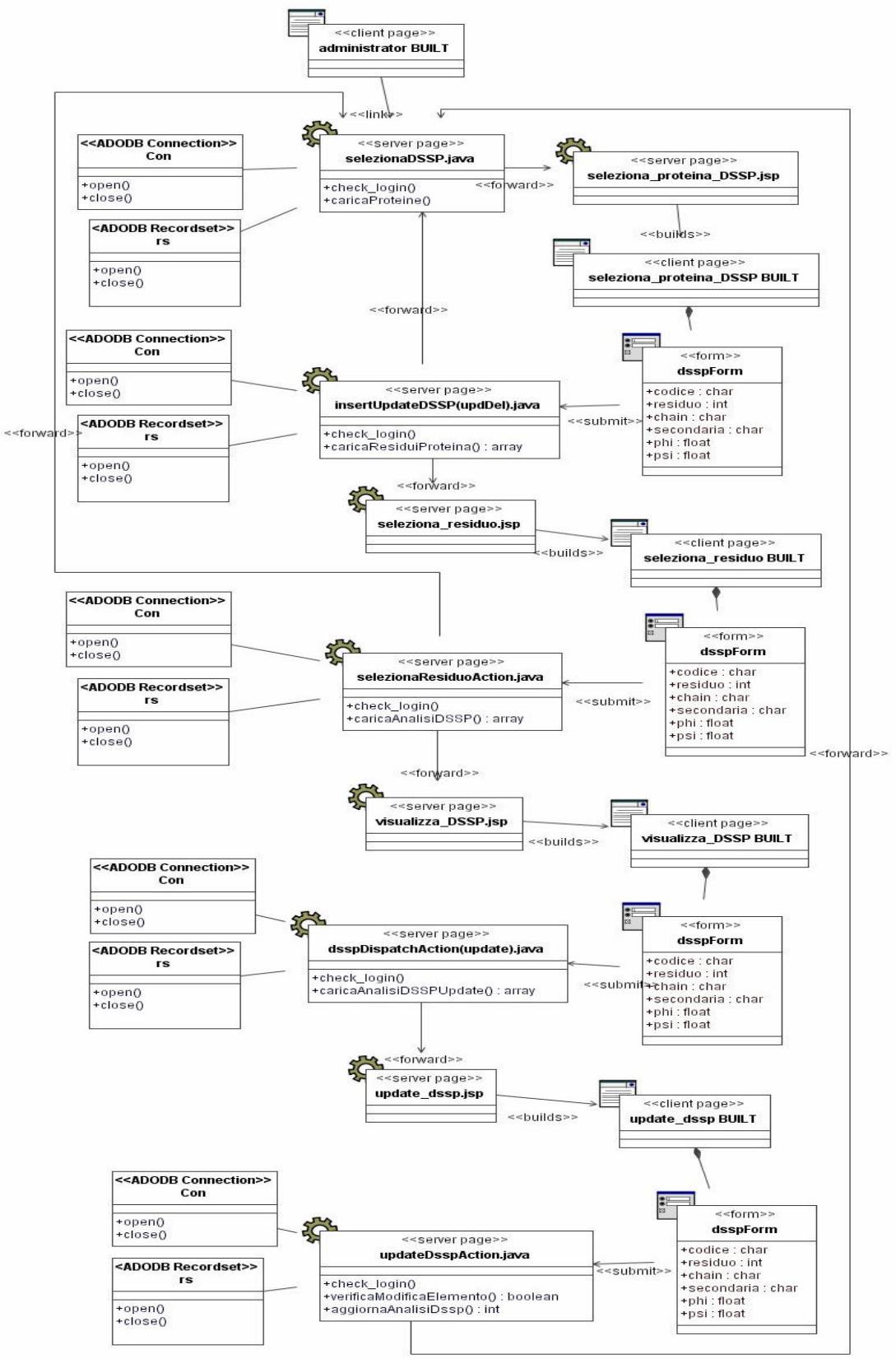
Caso d'uso: Visualizza Mutazione missense e Eliminazione Mutazione missense



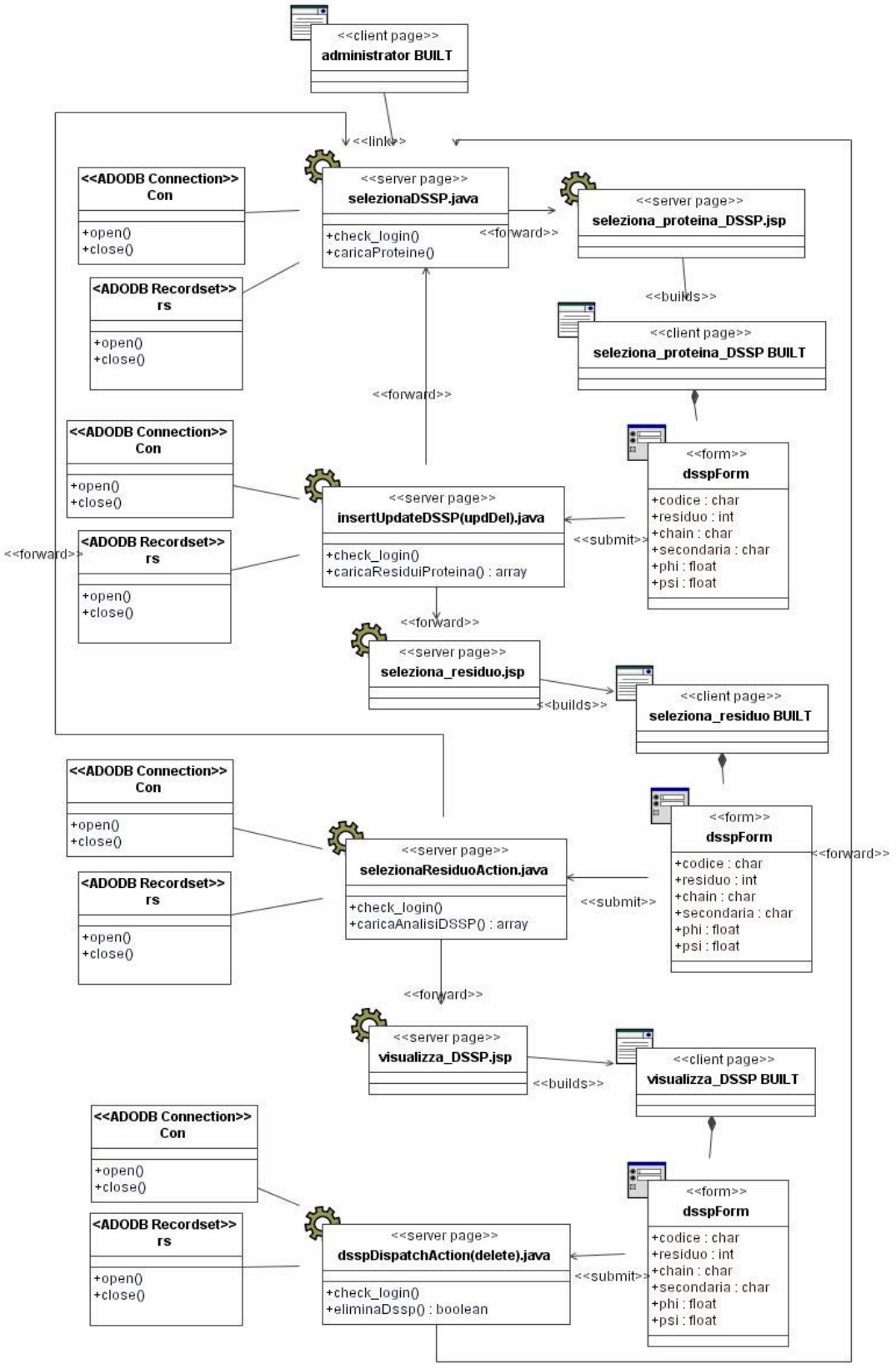
Caso d'uso: Inserimento Analisi DSSP



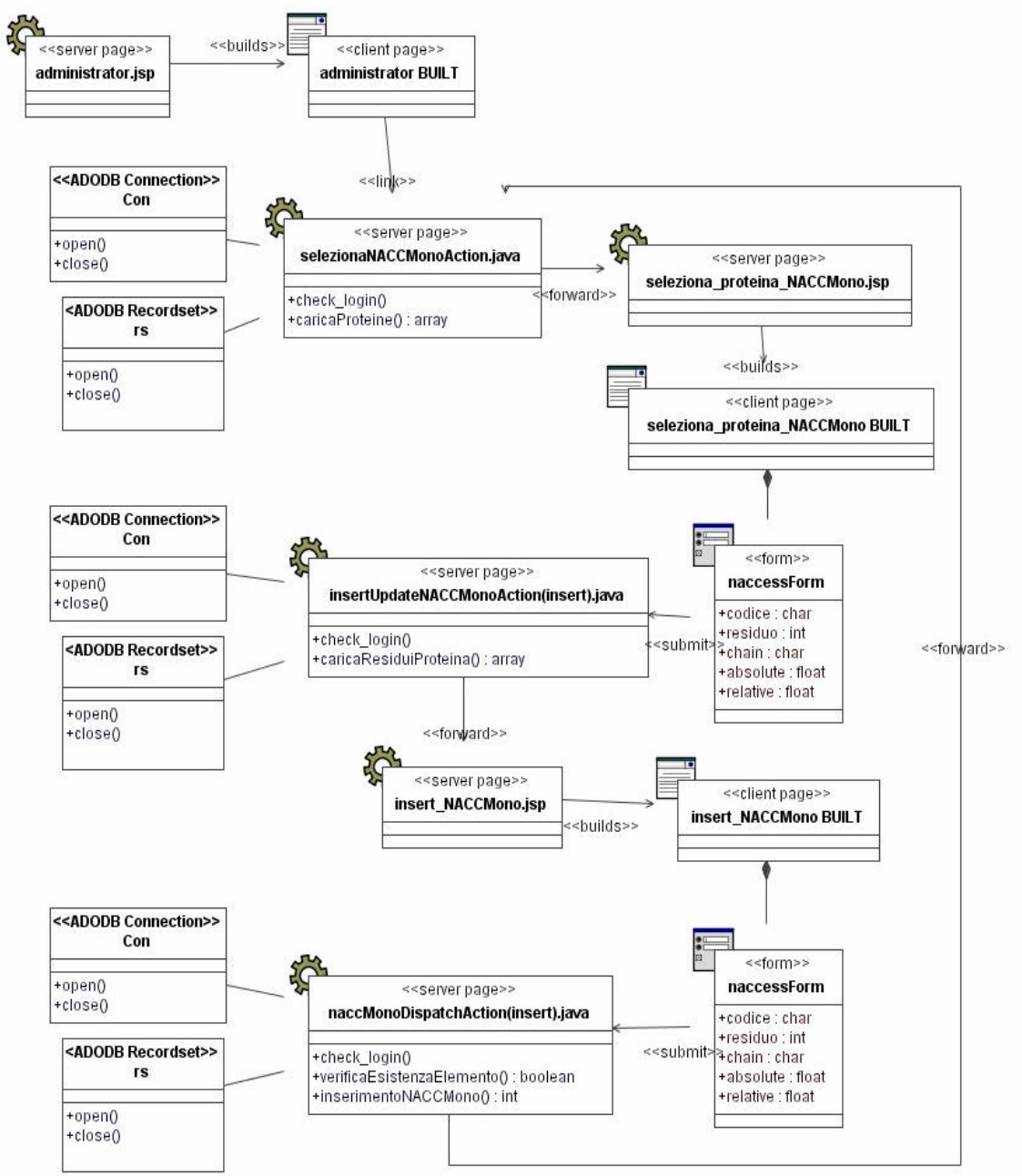
Caso d'uso: Visualizzazione Analisi DSSP e Modifica Analisi DSSP



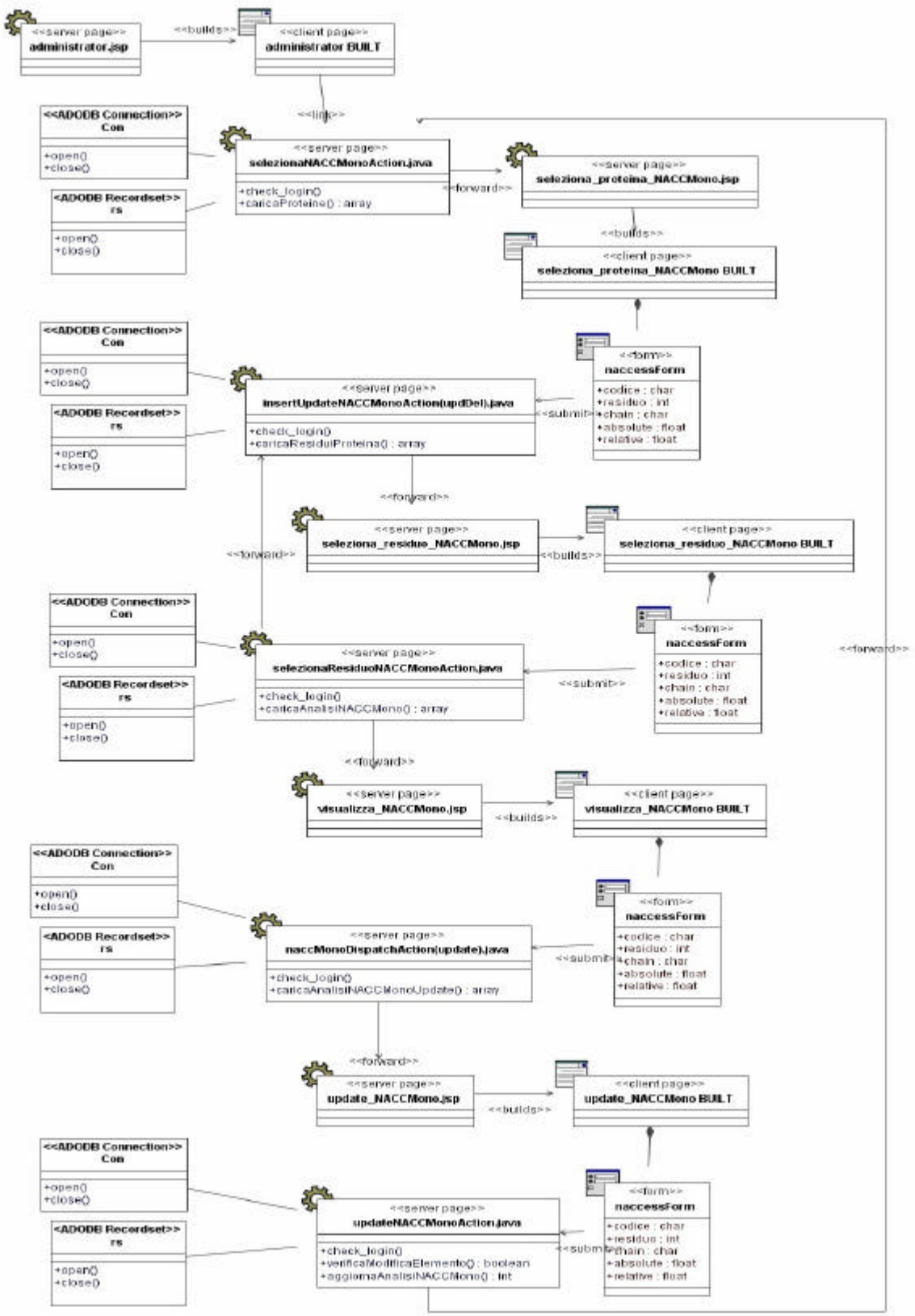
Caso d'uso: Visualizzazione Analisi DSSP e Eliminazione Analisi DSSP



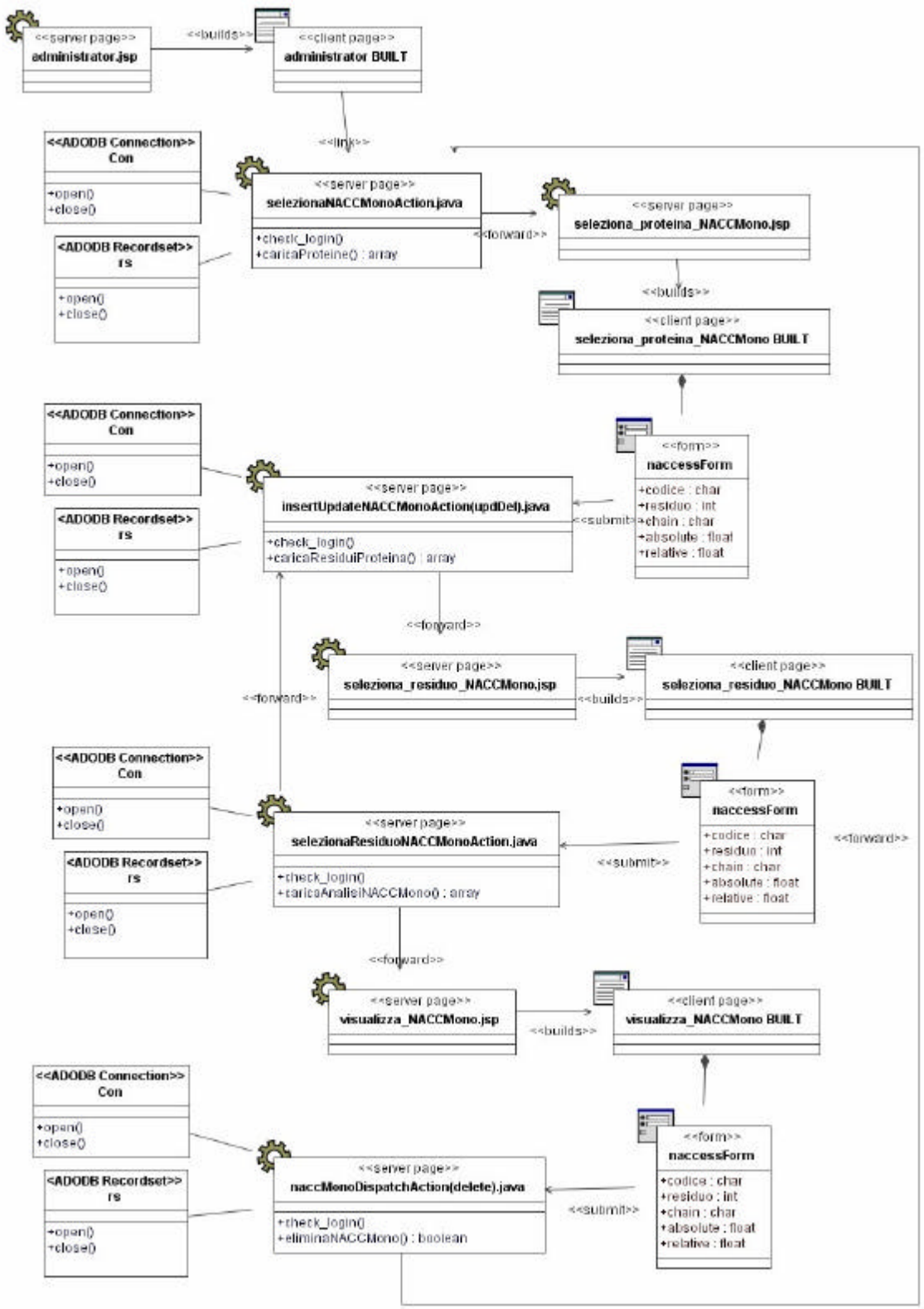
Caso d'uso: Inserimento Analisi NACCESS Monomers



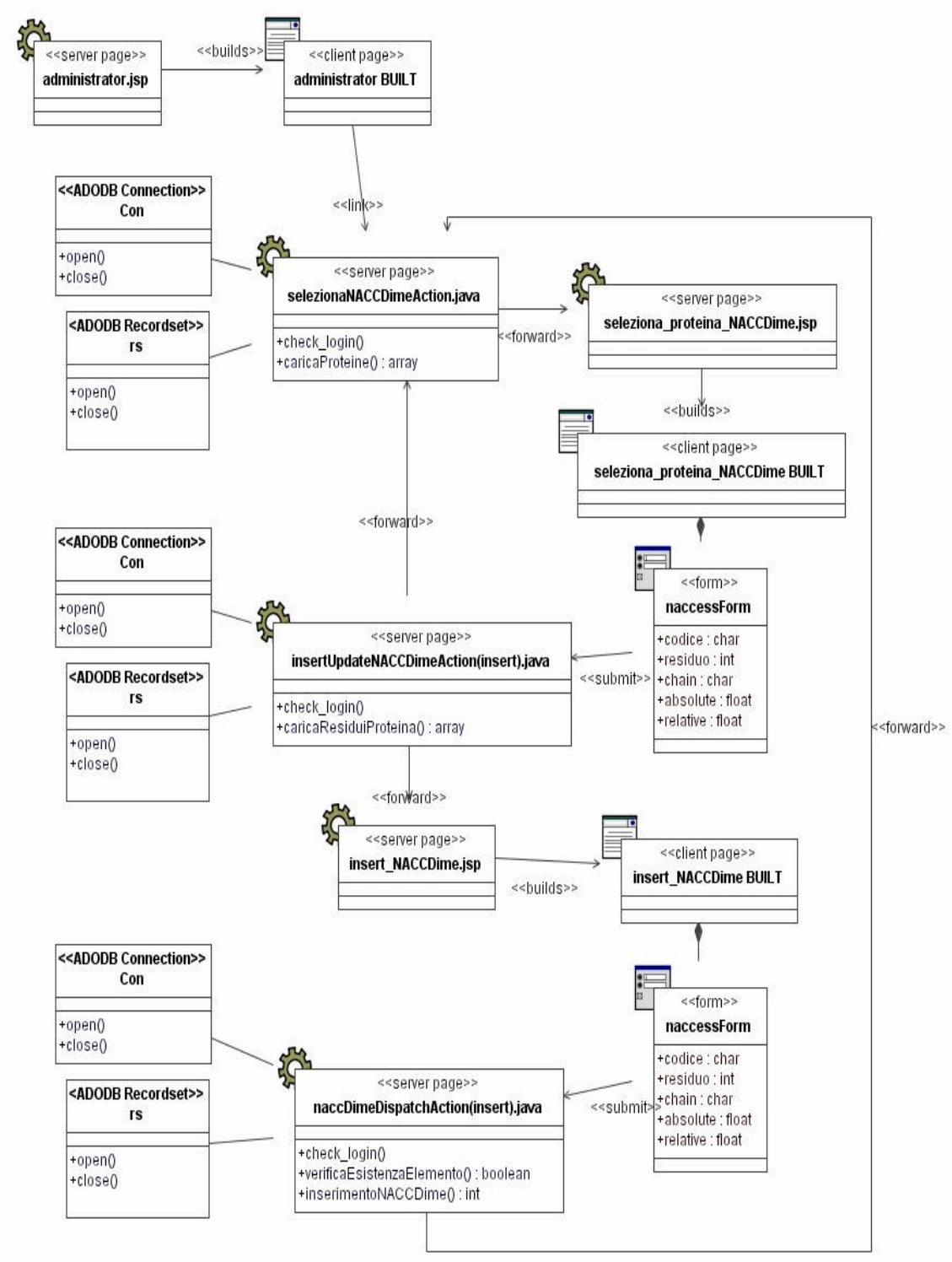
Caso d'uso: Visualizzazione Analisi NACCESS Monomers e Modifica Analisi NACCESS Monomers



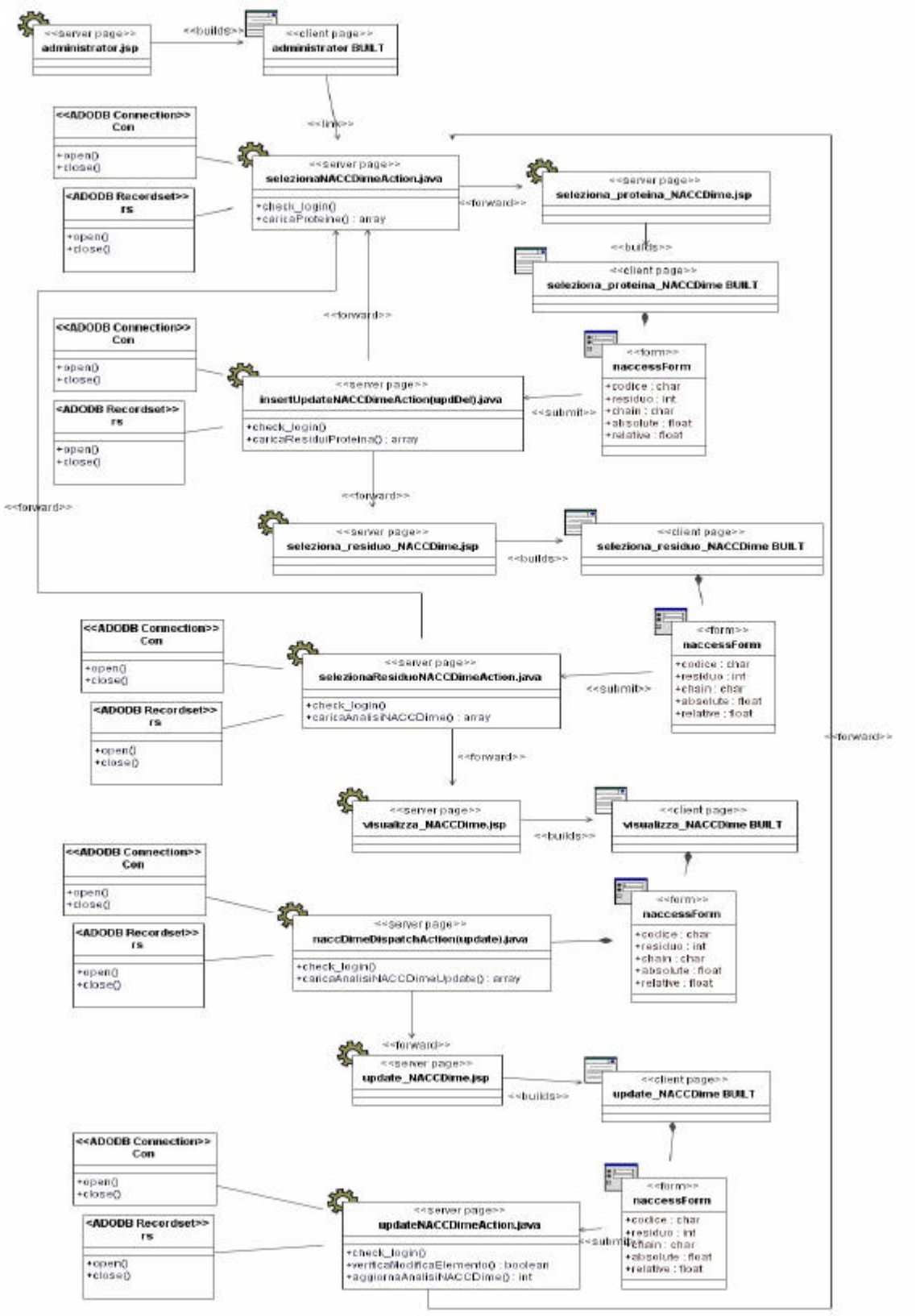
Caso d'uso: Visualizzazione Analisi NACCESS Monomers e Eliminazione Analisi NACCESS Monomers



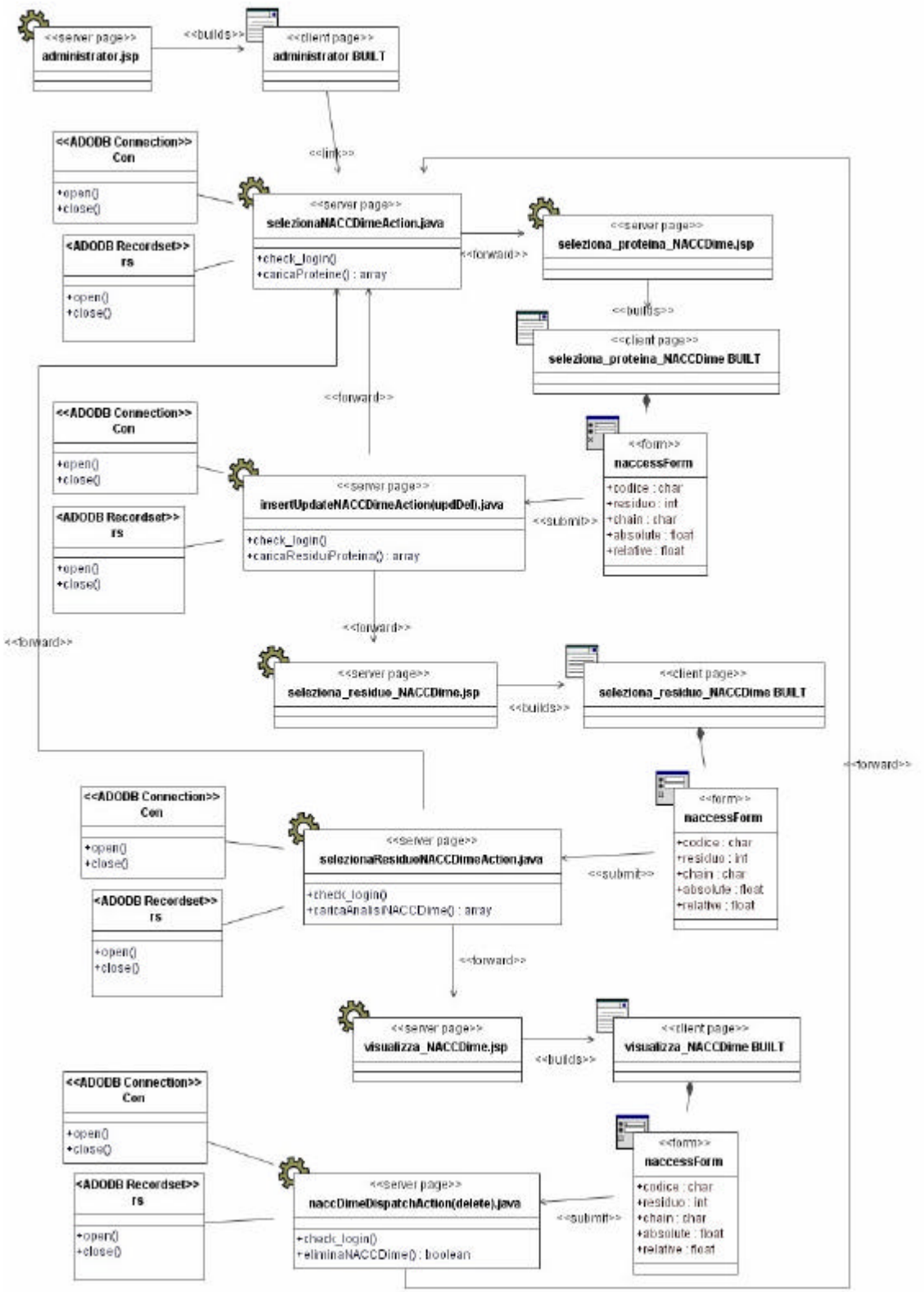
Caso d'uso: Inserimento Analisi NACCESS Dimer



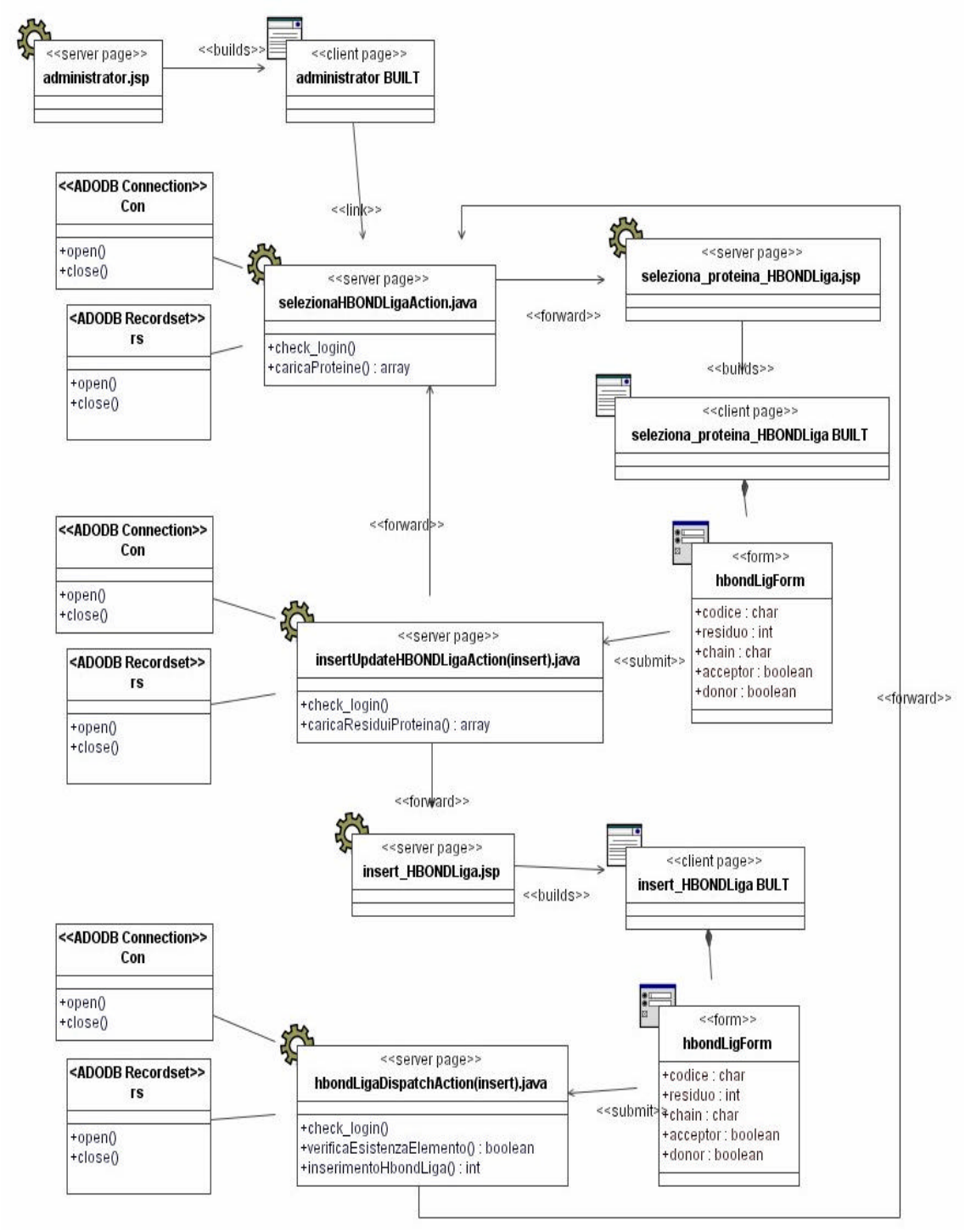
Caso d'uso: Visualizzazione Analisi NACCESS Dimer e Modifica Analisi NACCESS Dimer



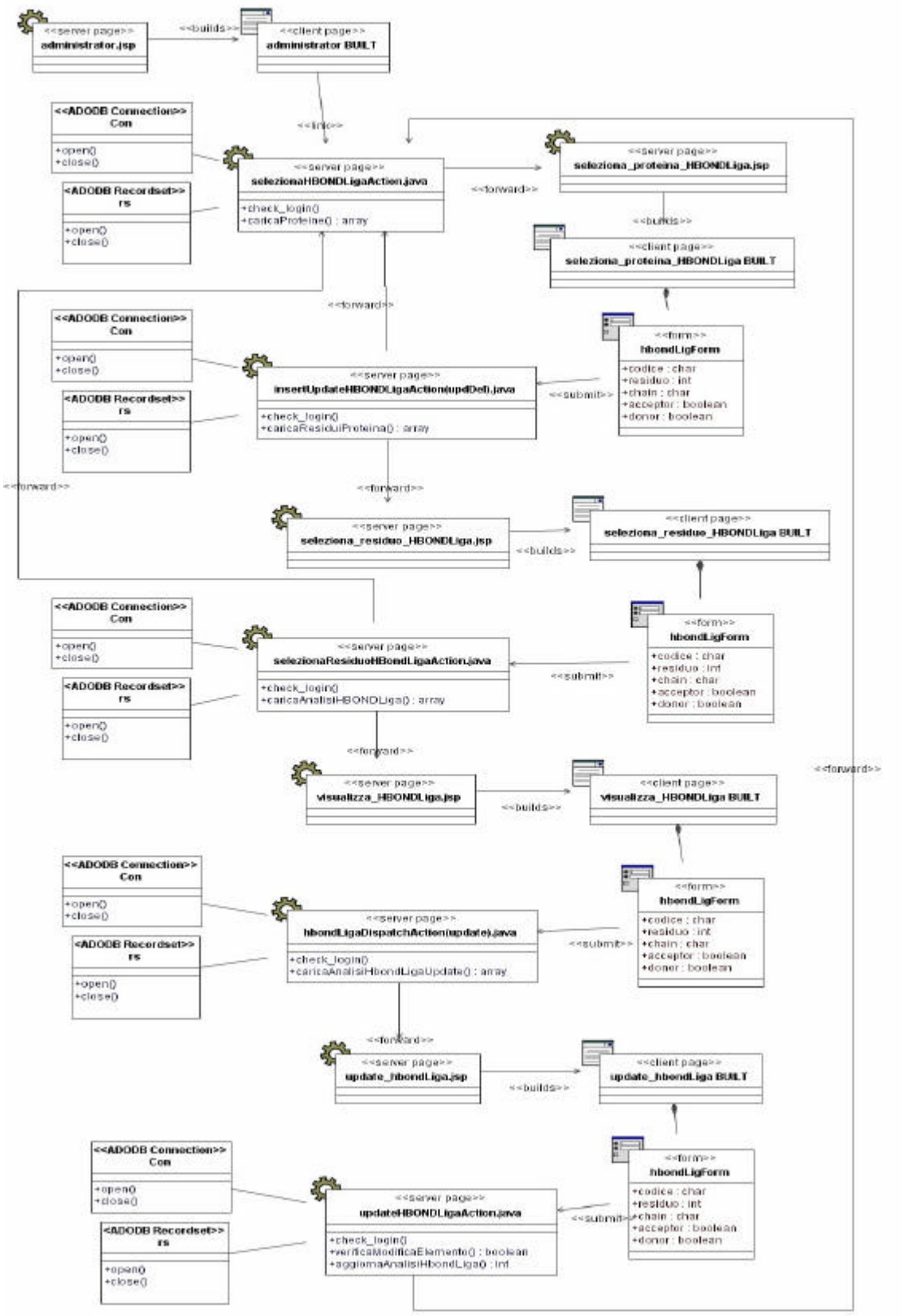
Caso d'uso: Visualizzazione Analisi NACCESS Dimer e Eliminazione Analisi NACCESS Dimer



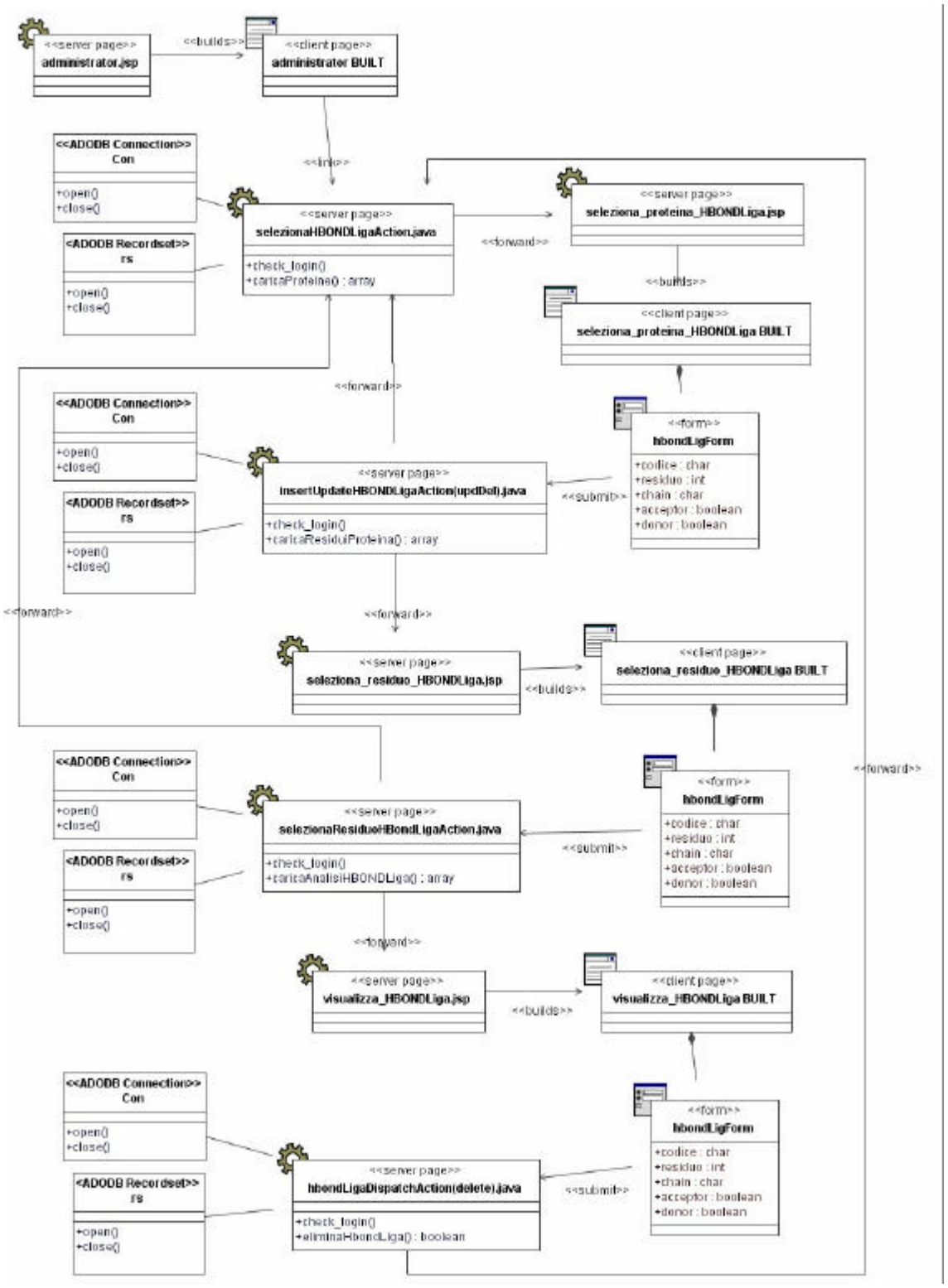
Caso d'uso: Inserimento Analisi H-BOND con Ligando



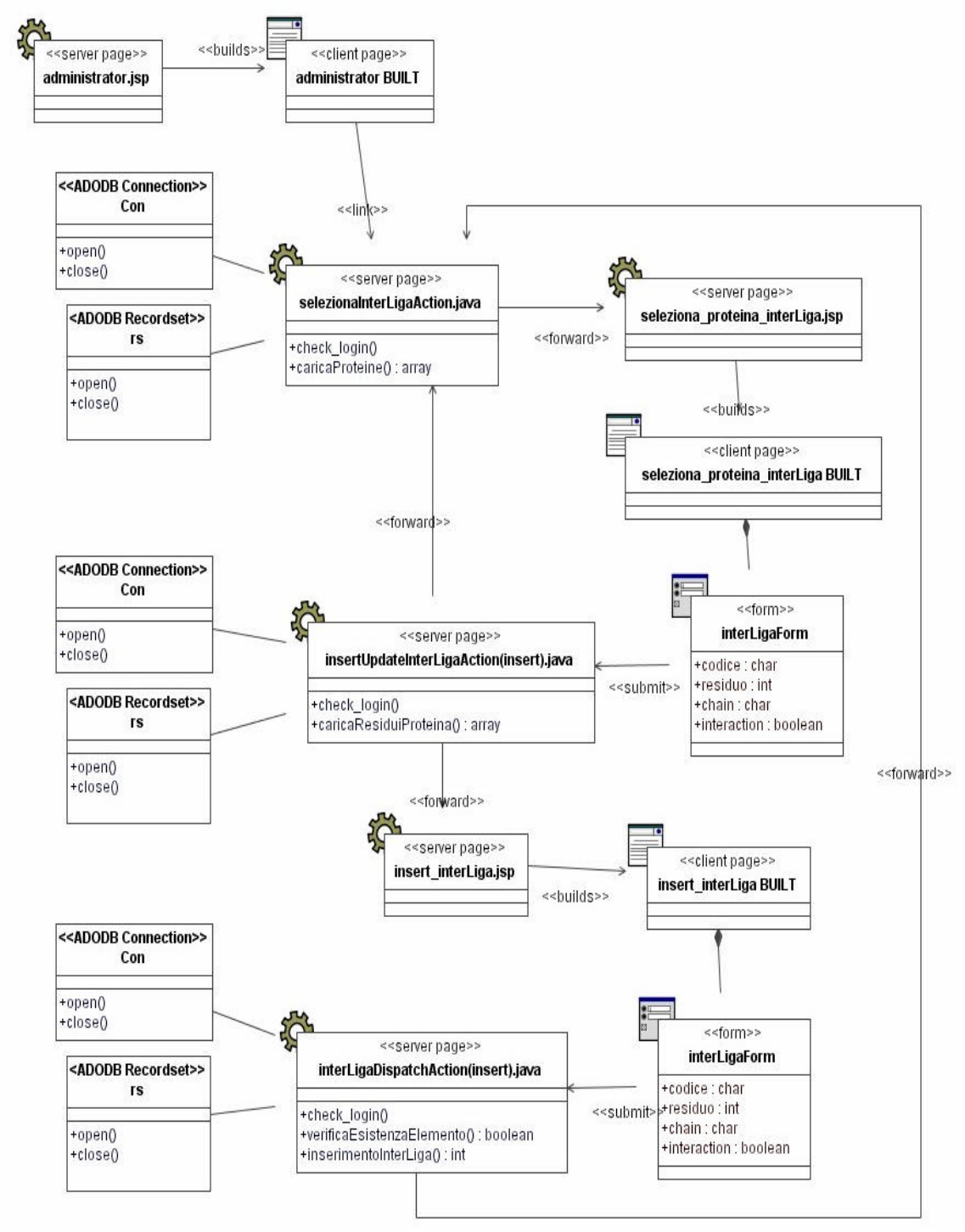
Caso d'uso: Visualizzazione Analisi H-BOND con Ligando e Modifica Analisi H-BOND con Ligando



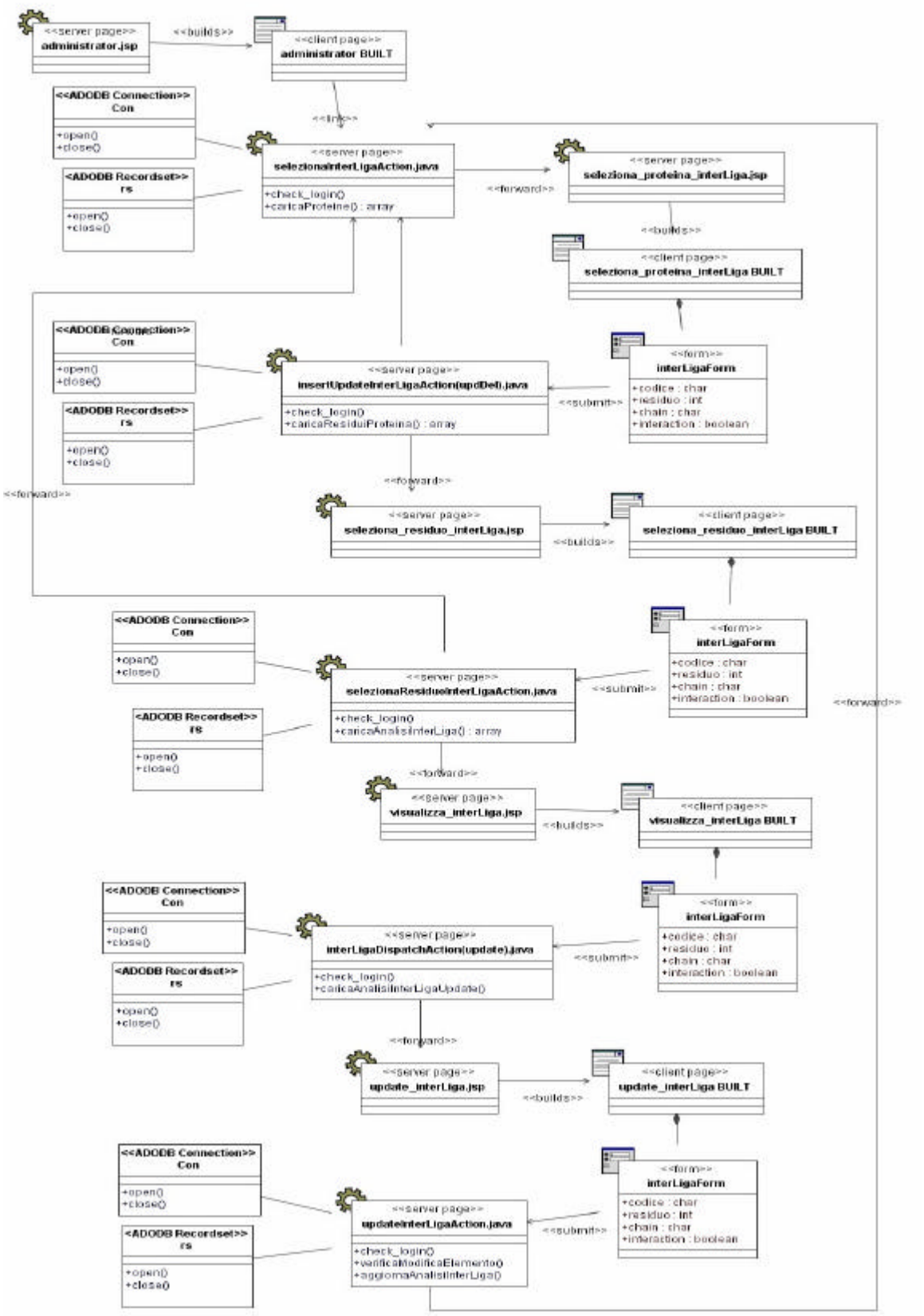
Caso d'uso: Visualizzazione Analisi H-BOND con Ligando e Eliminazione Analisi H-BOND con Ligando



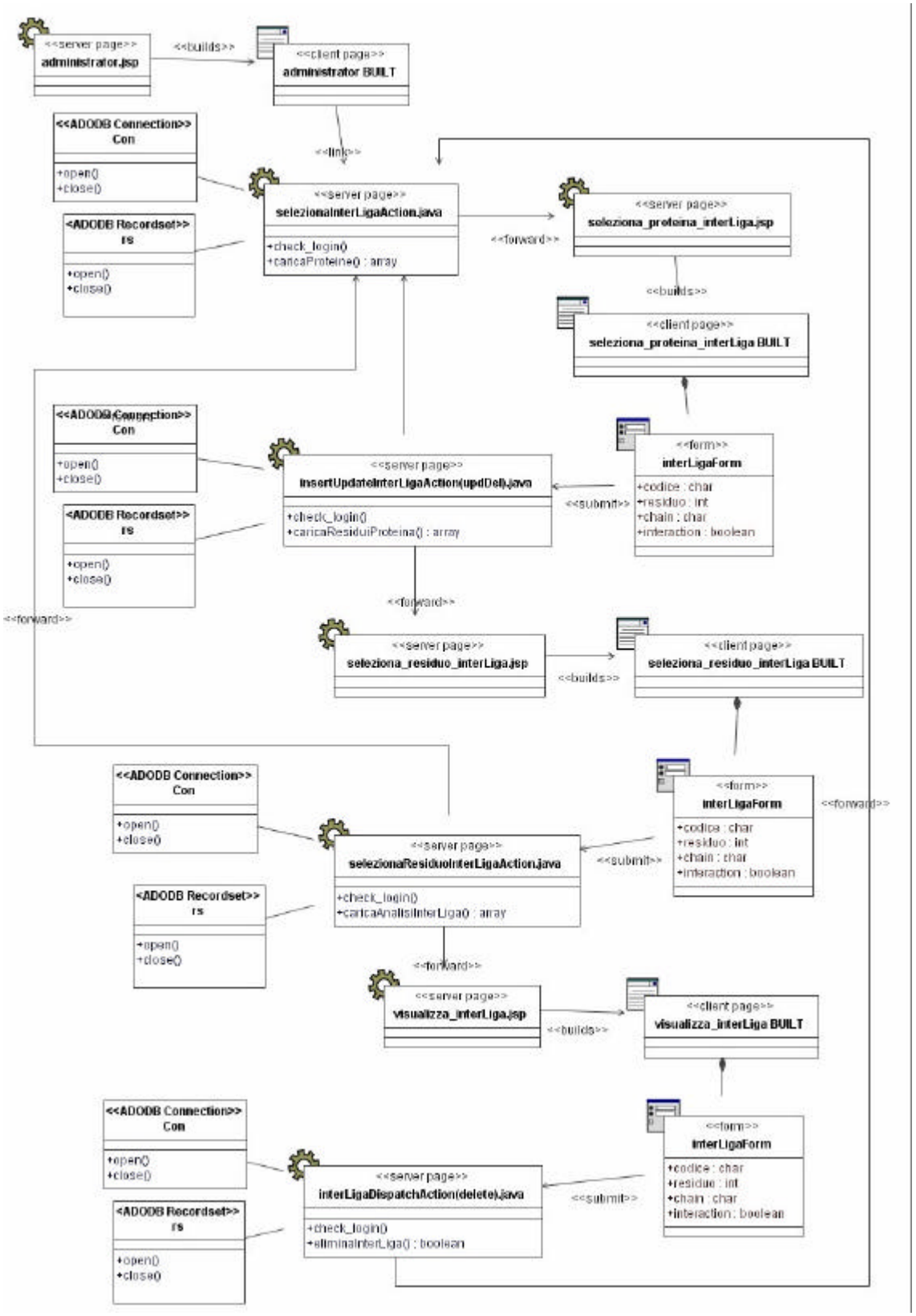
Caso d'uso: Inserimento Analisi Interazioni con Ligando



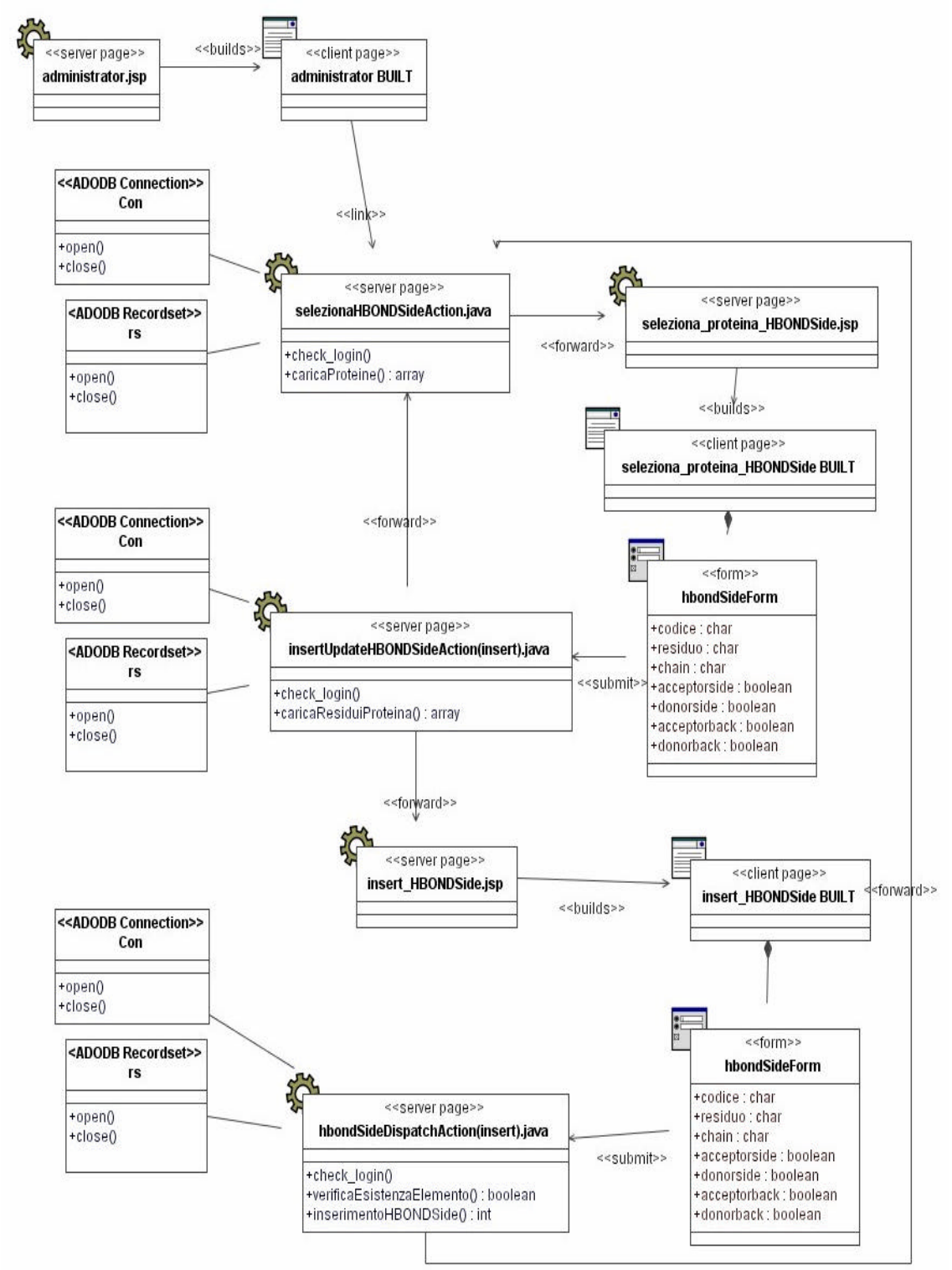
Caso d'uso: Visualizzazione Analisi Interazioni con Ligando e Modifica Analisi Interazioni con Ligando



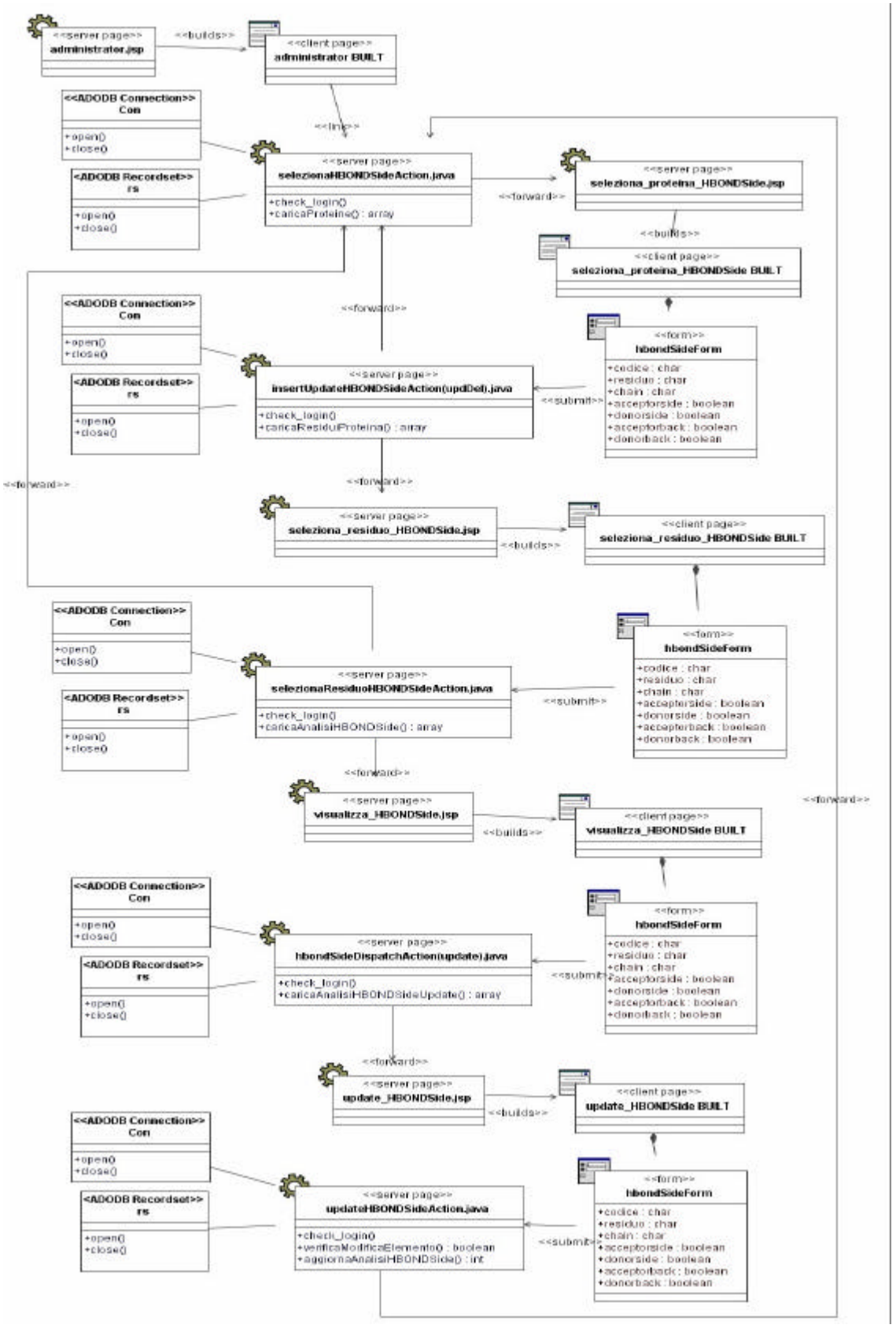
Caso d'uso: Visualizzazione Analisi Interazioni con Ligando e Eliminazione Analisi Interazioni con Ligando



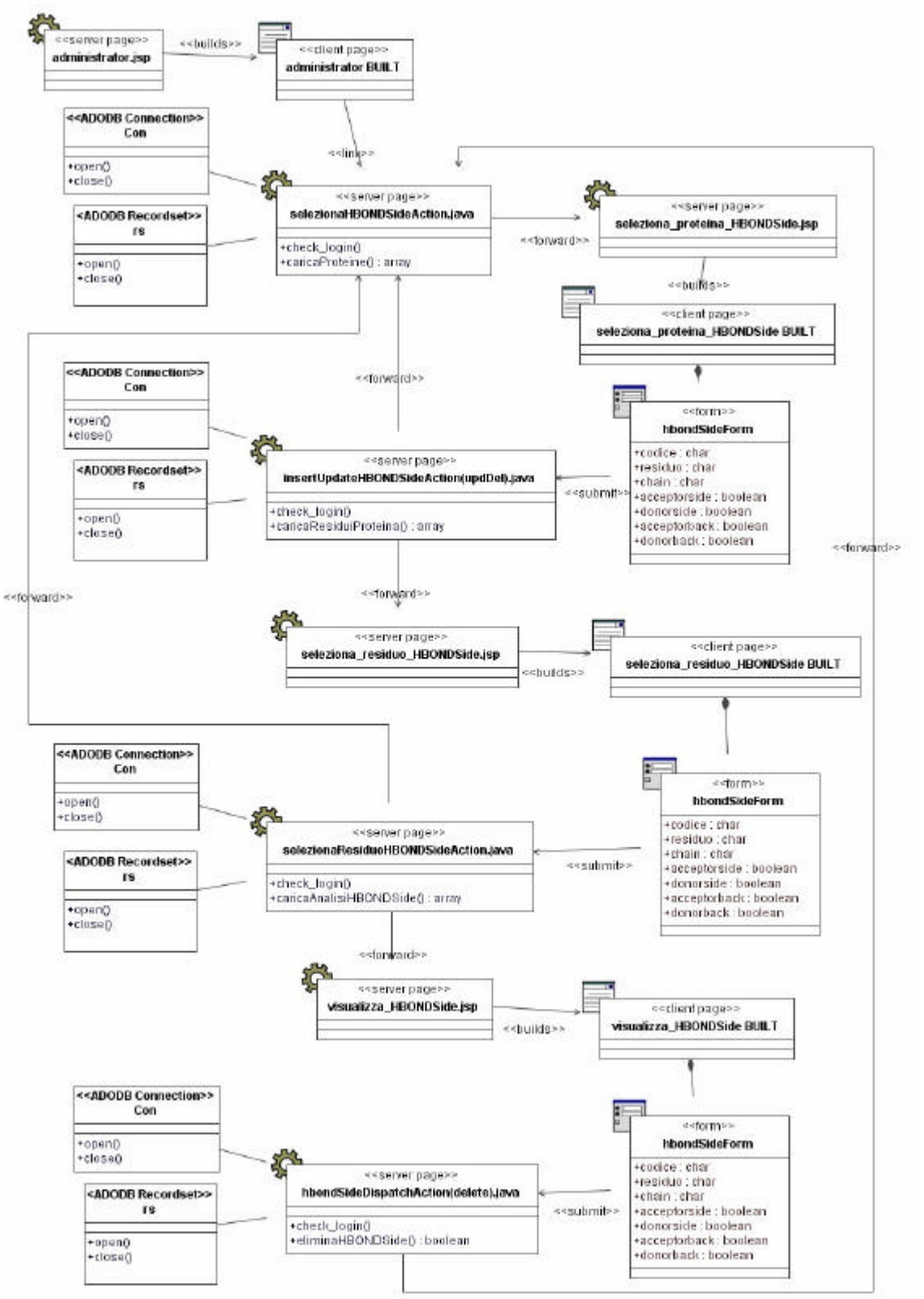
Caso d'uso: Inserimento Analisi H-BOND con Sidechains



Caso d'uso: Visualizzazione Analisi H-BOND con SideChains e Modifica Analisi H-BOND con SideChains

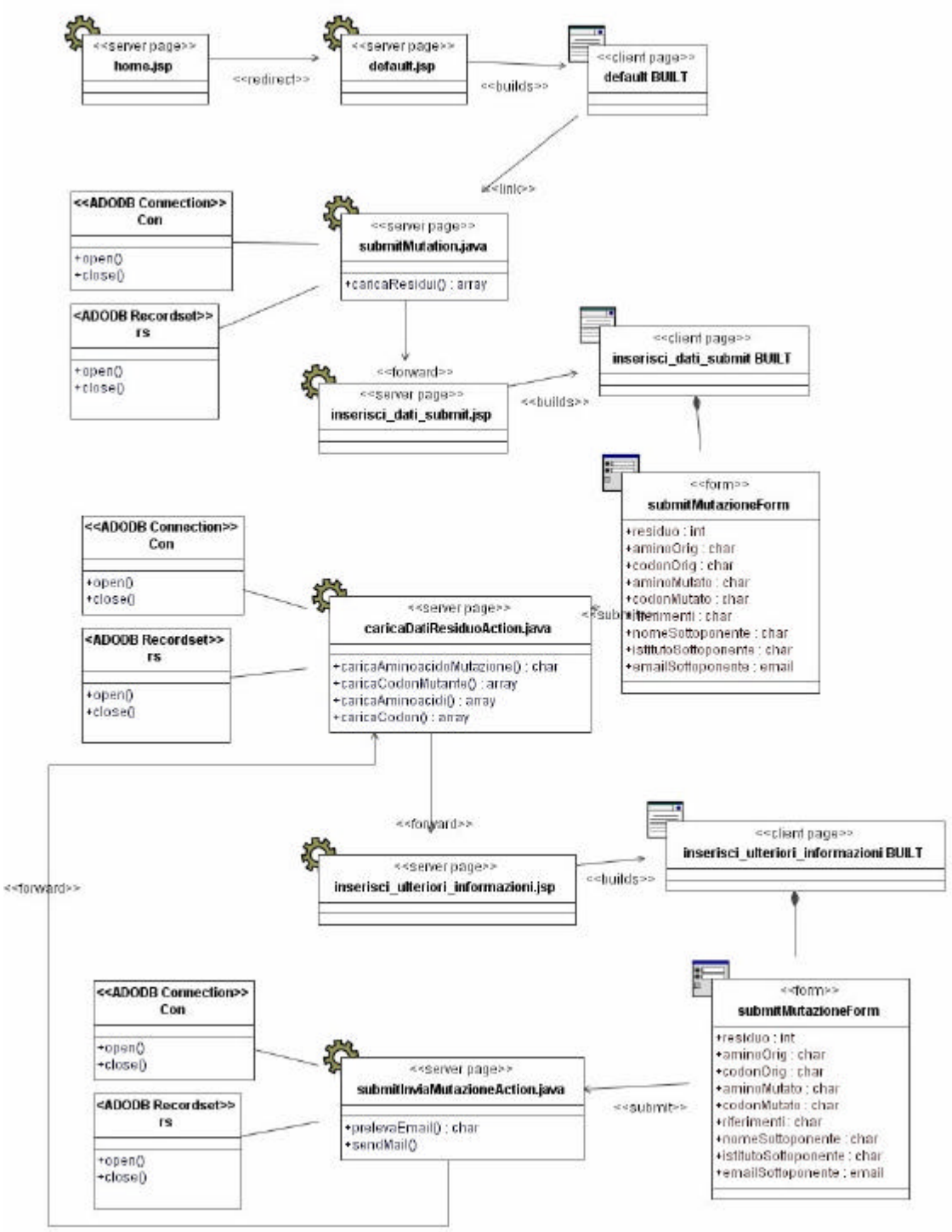


Caso d'uso: Visualizzazione Analisi H-BOND con SideChains e Eliminazione Analisi H-BOND con SideChains

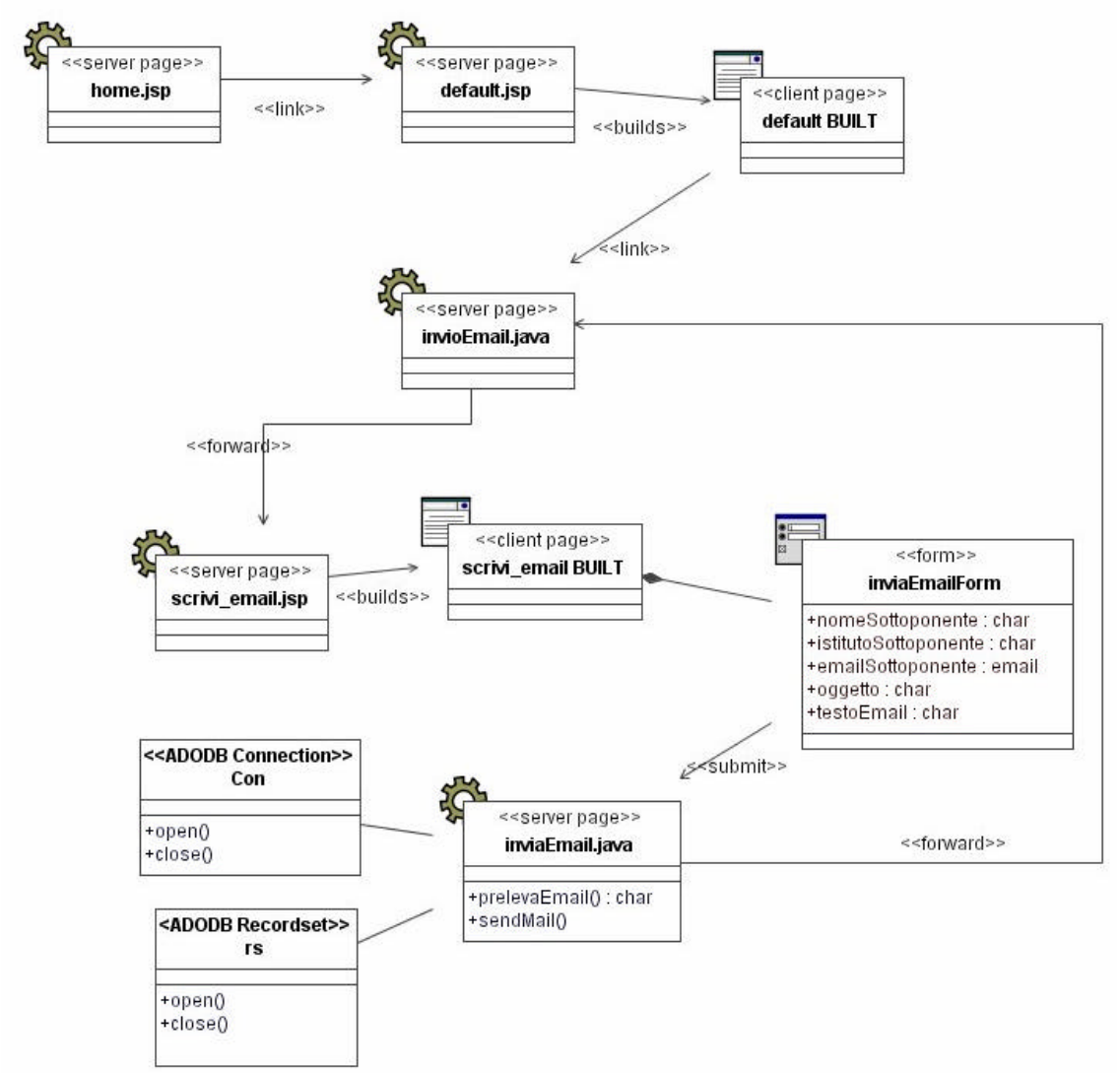


❖ Modulo Pubblico

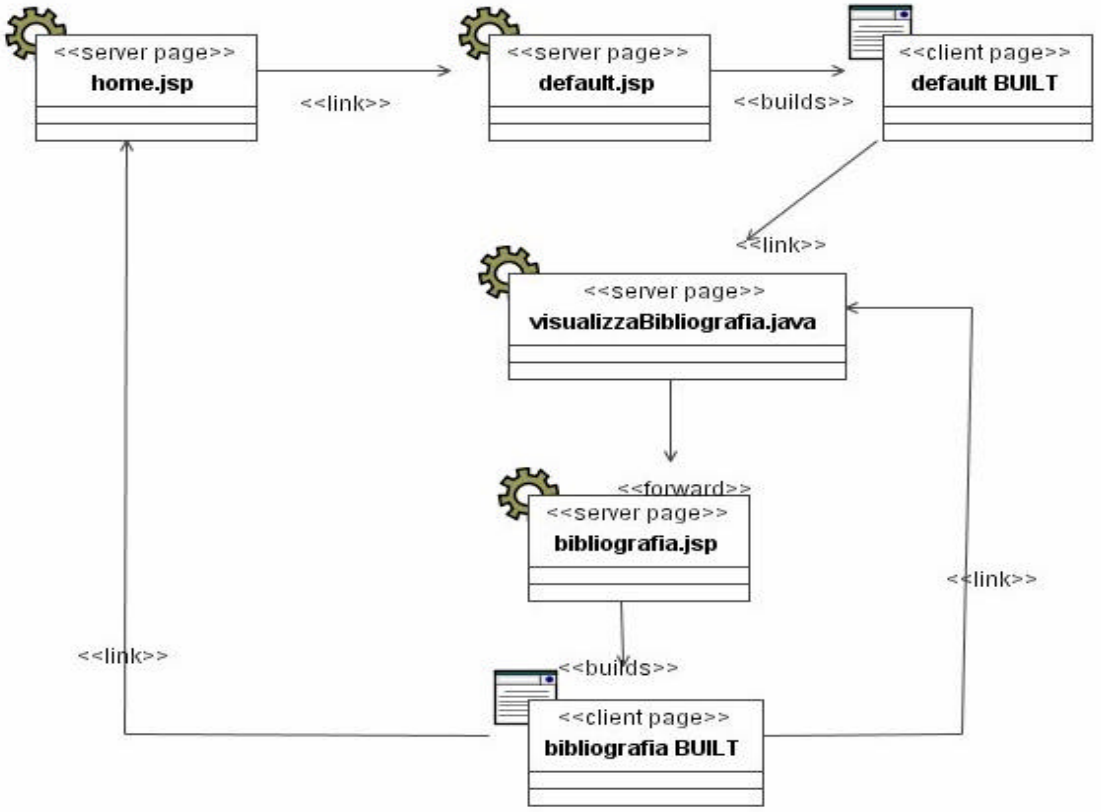
Caso d'uso: Submit nuova mutazione



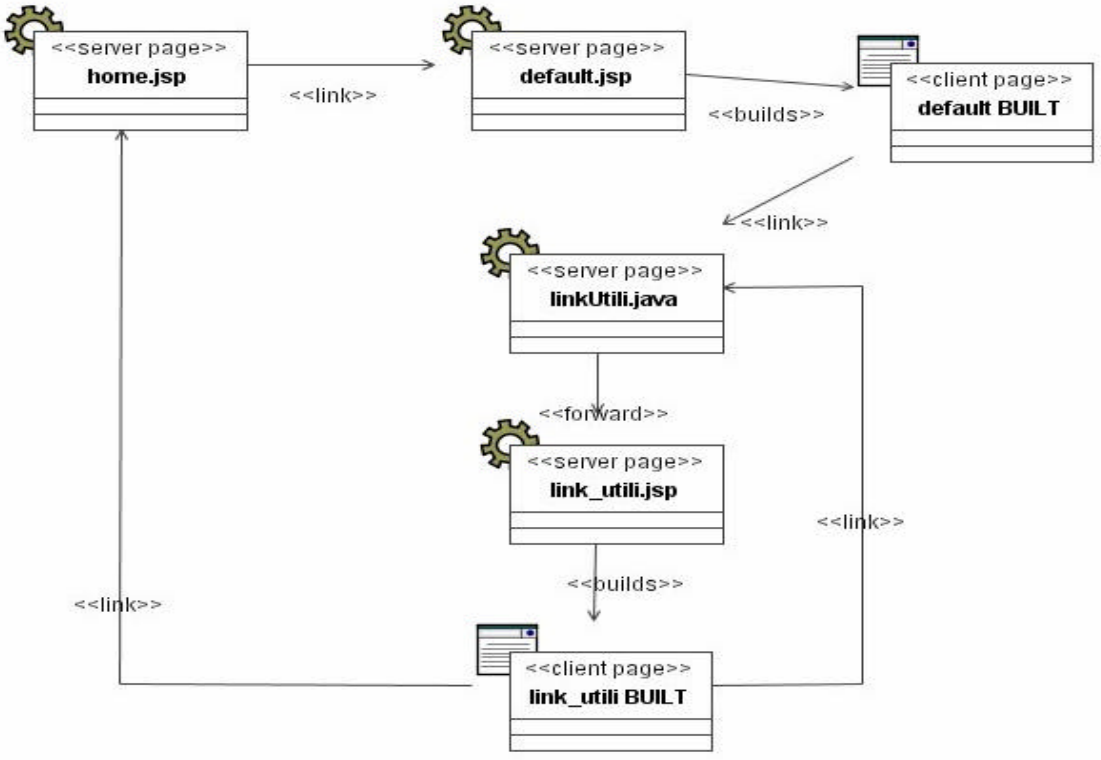
Caso d'uso: Invia e-mail all'amministratore



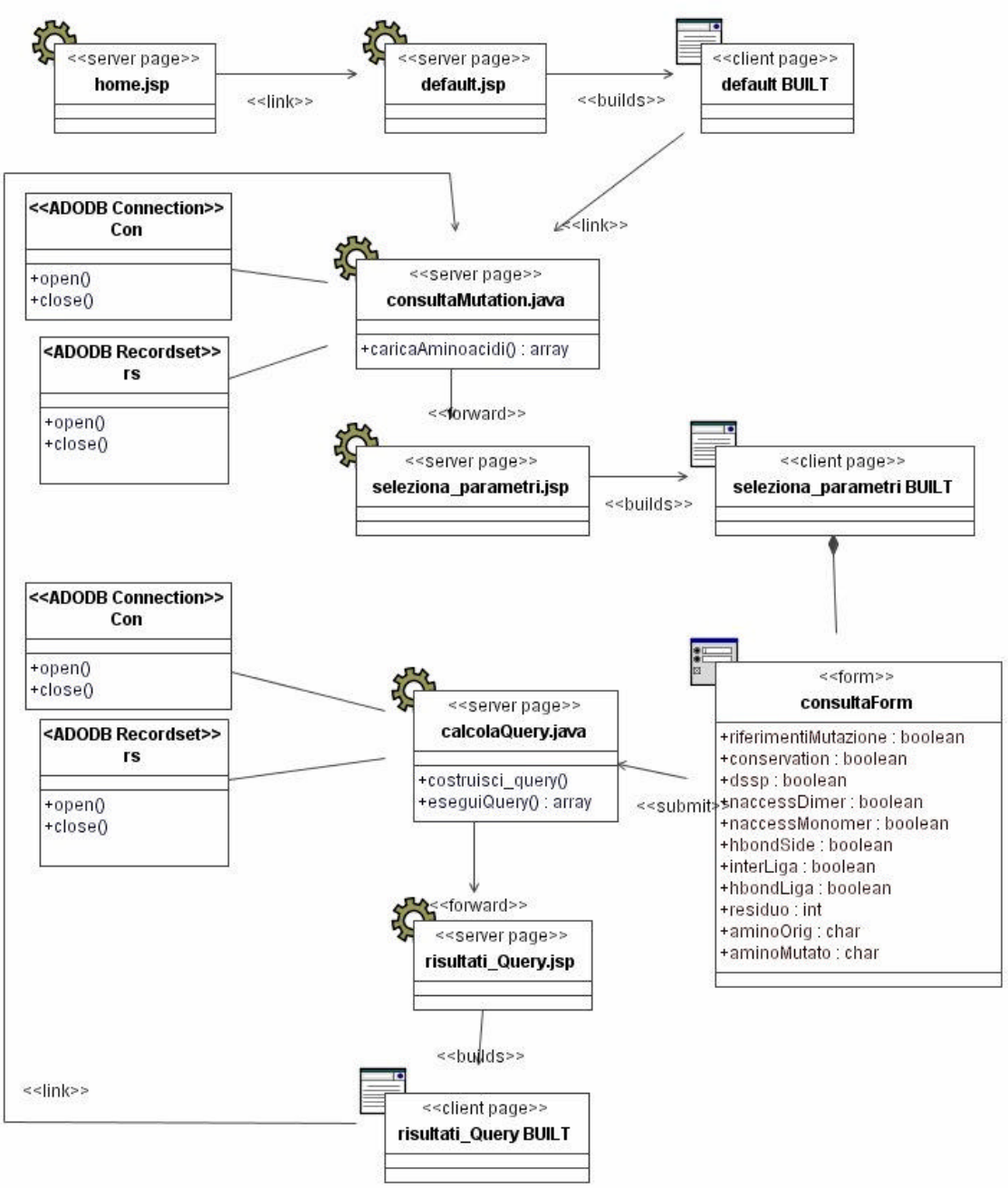
Caso d'uso: Visualizza bibliografia



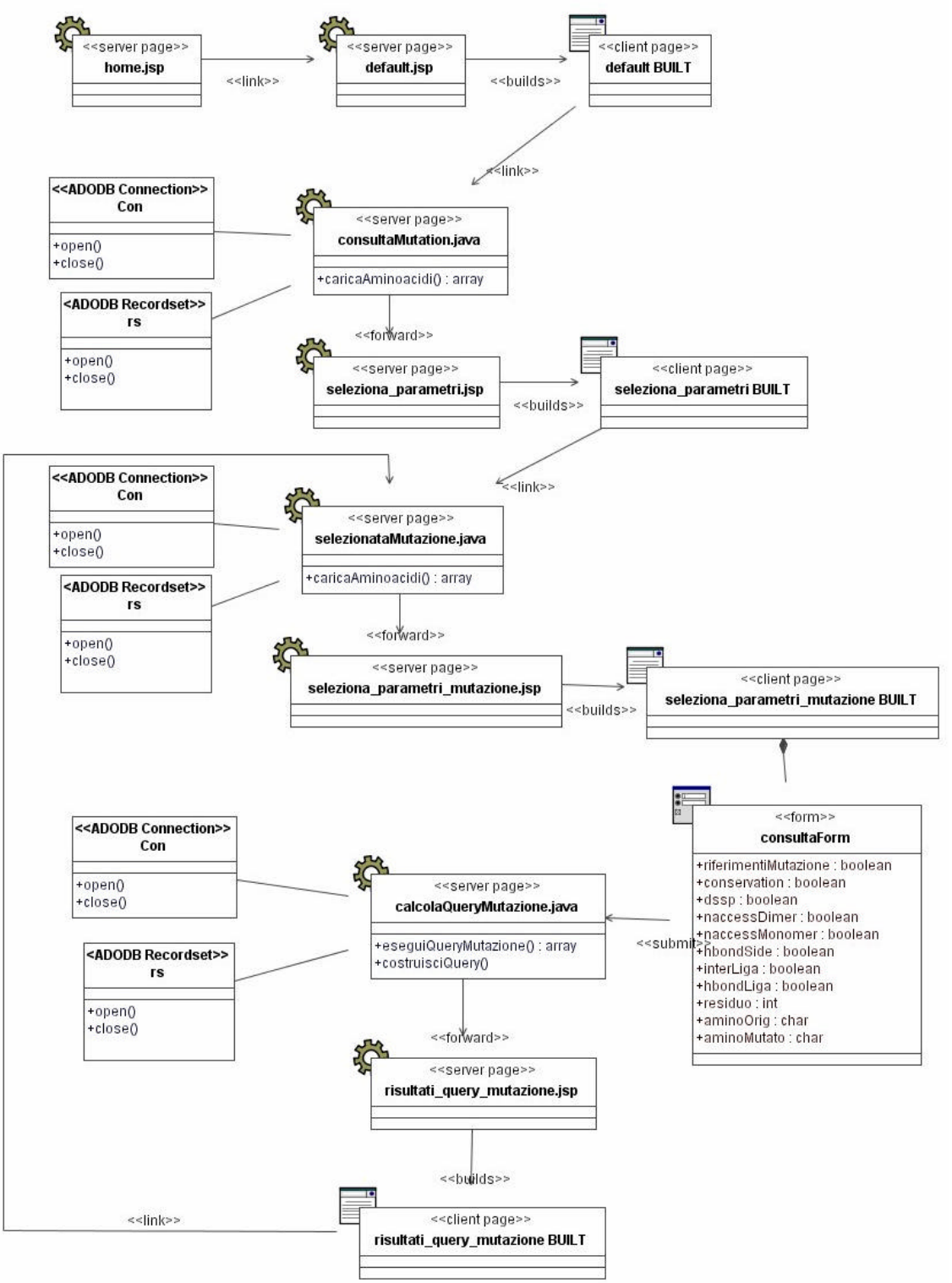
Caso d'uso: Visualizza link utili



Caso d'uso: Consultazione informazioni sull'intera proteina



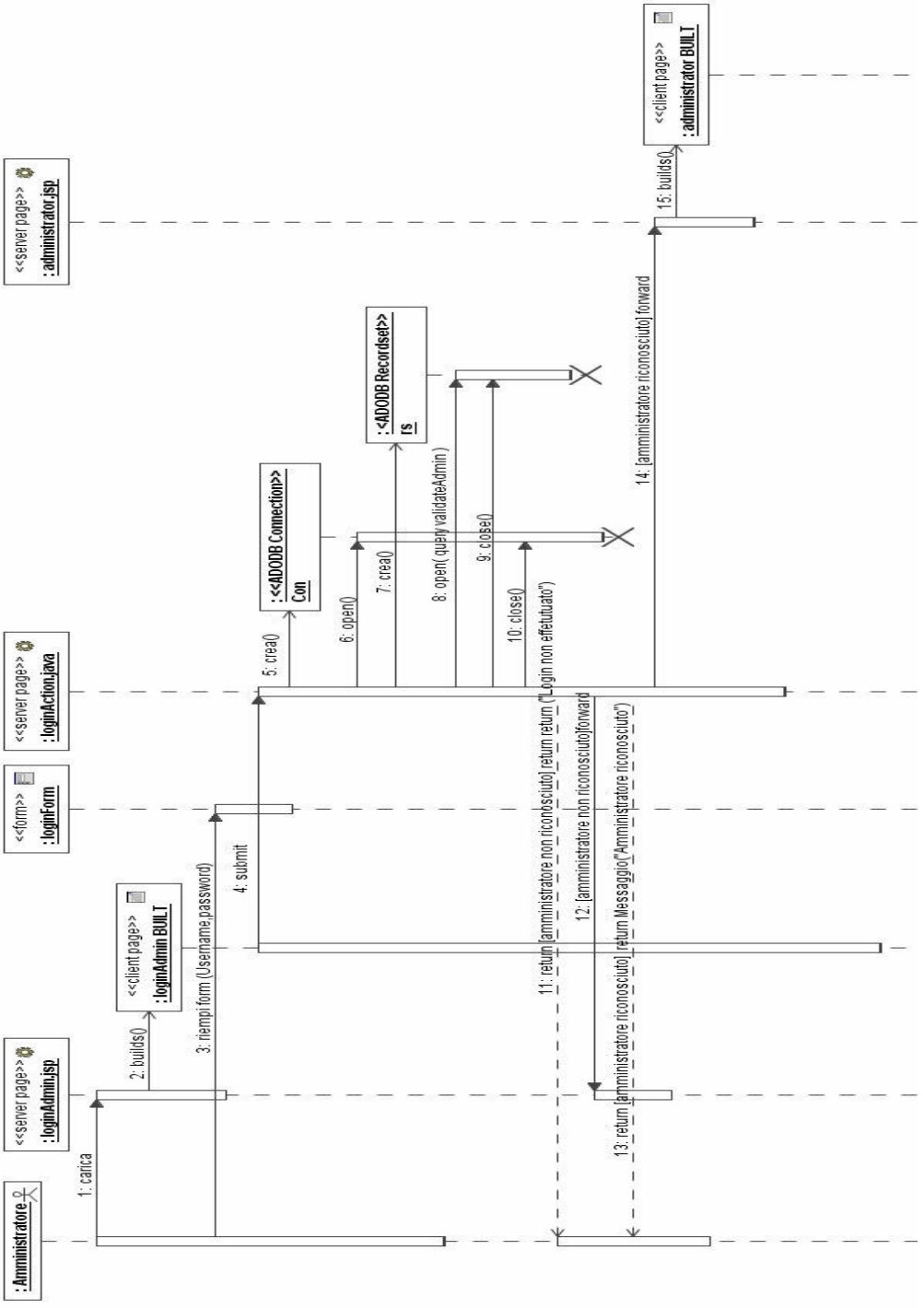
Caso d'uso: Consultazione informazioni sulle mutazioni



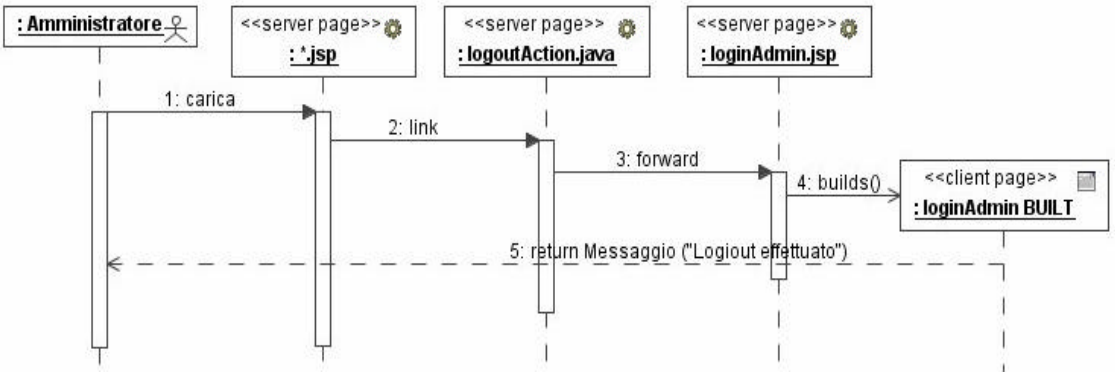
B.4 Sequence Diagrams

❖ Modulo Amministrazione

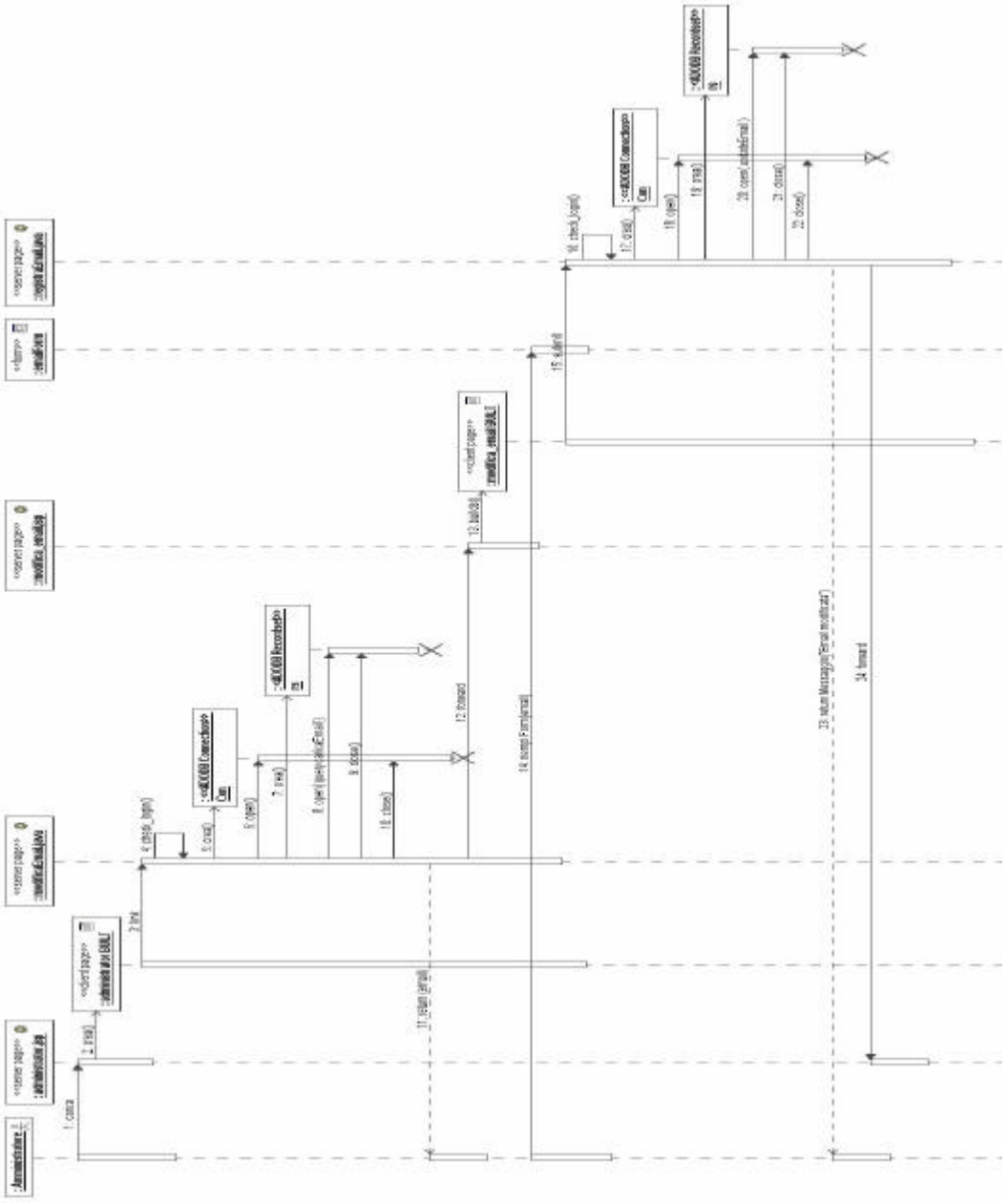
Caso d'uso: Login



Caso d'uso: Logout

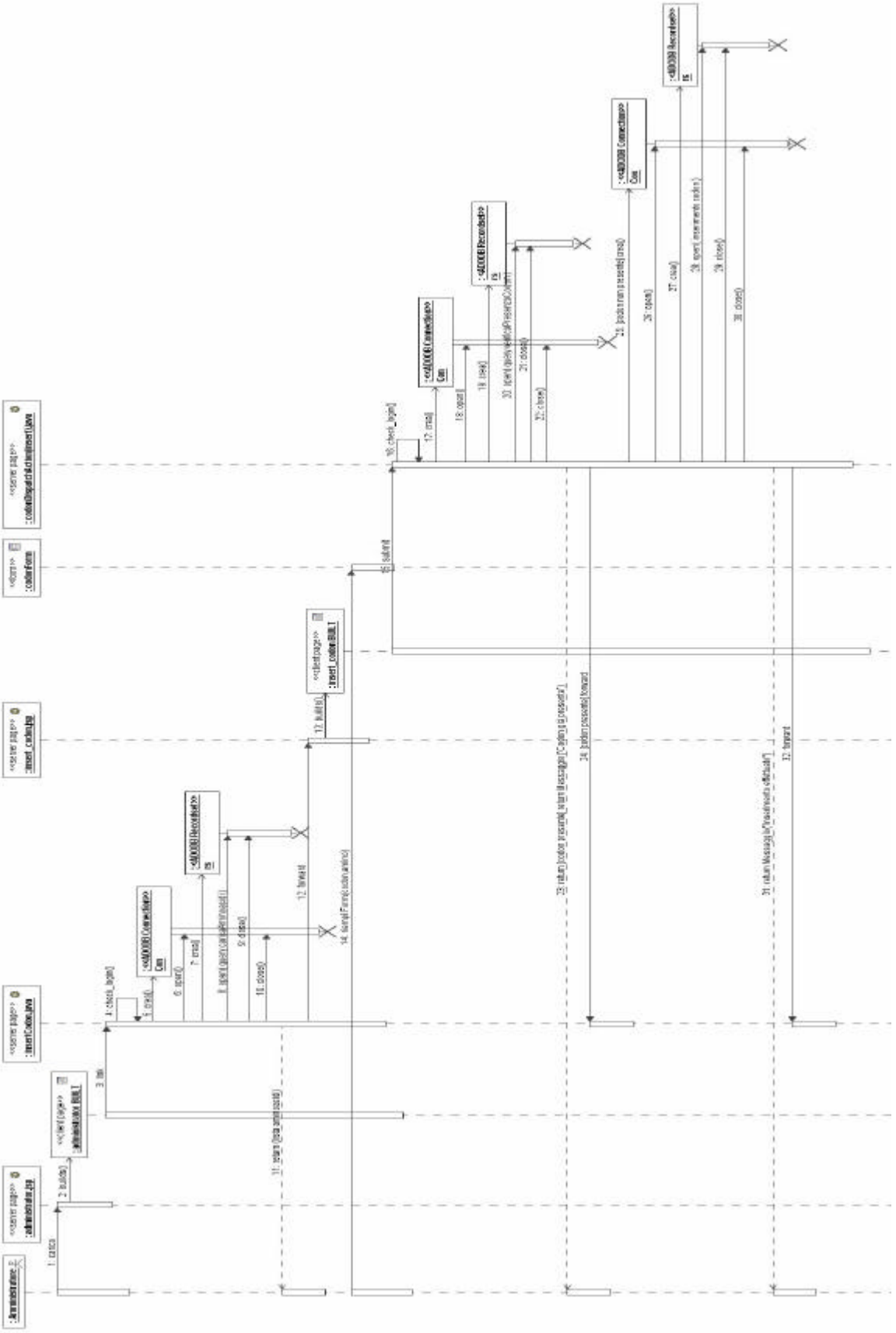


Caso d'uso: modifica email amministratore

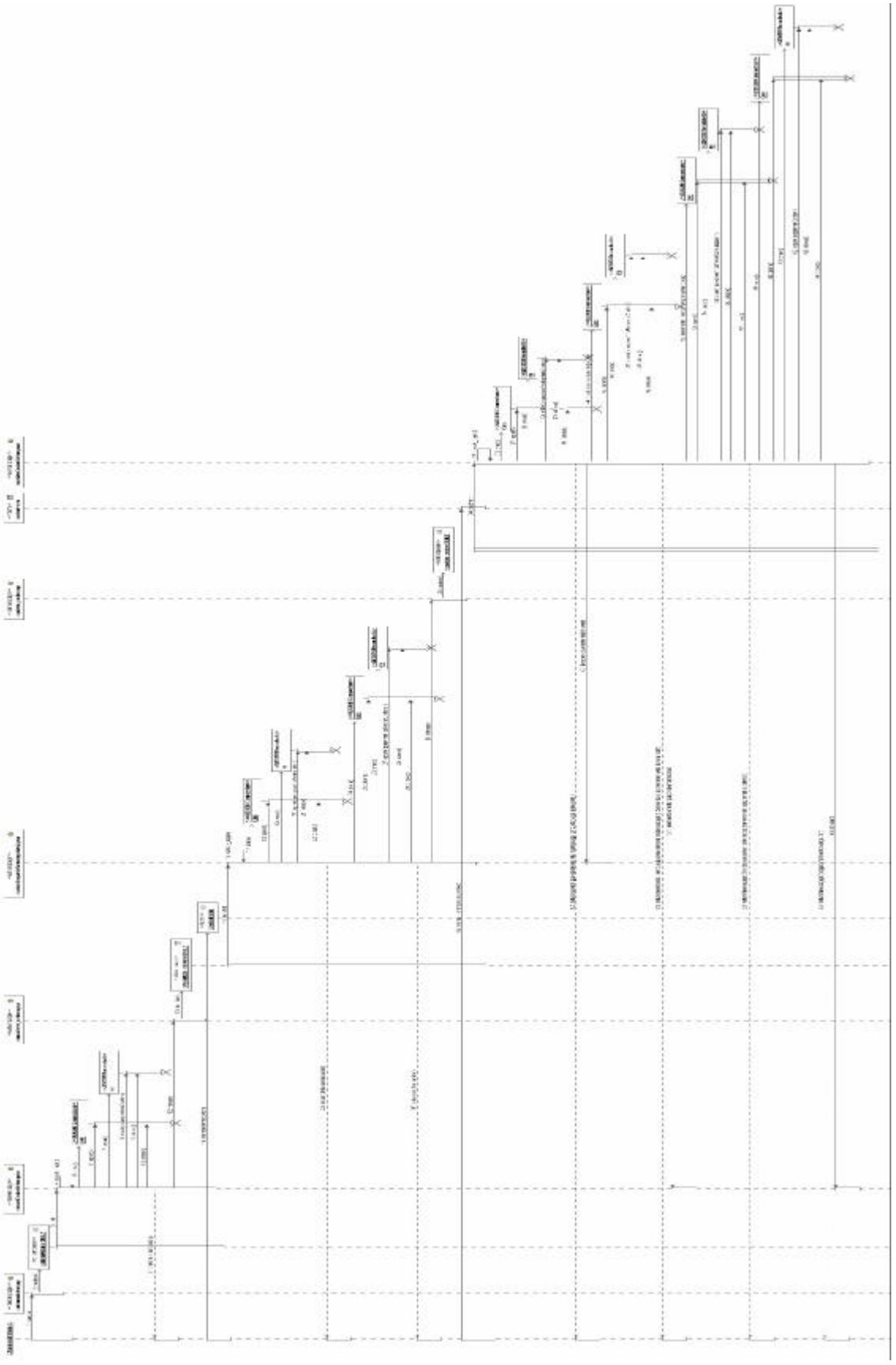


Caso d'uso: modifica username/password di amministrazione

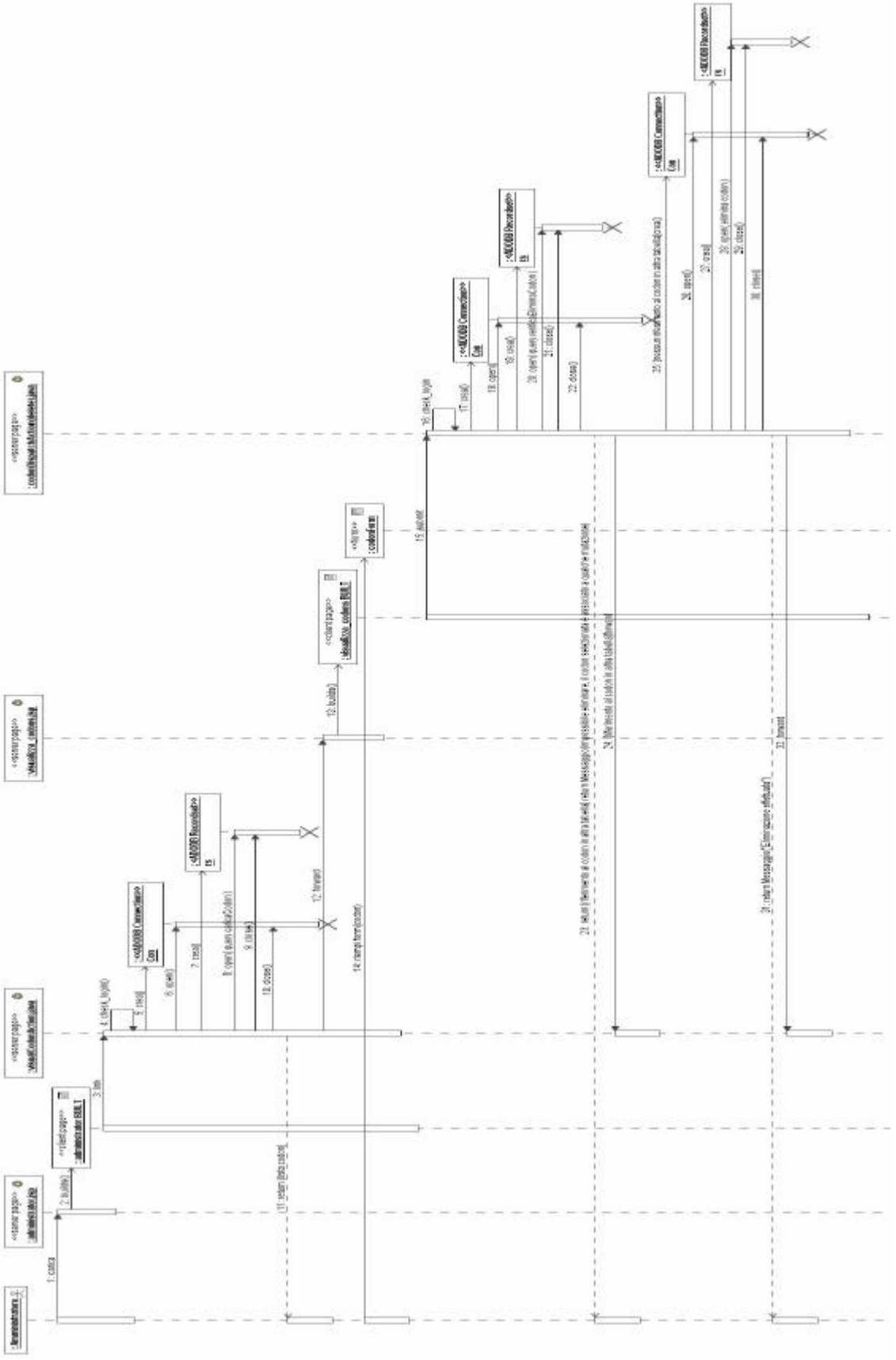
Caso d'uso: Inserimento codon



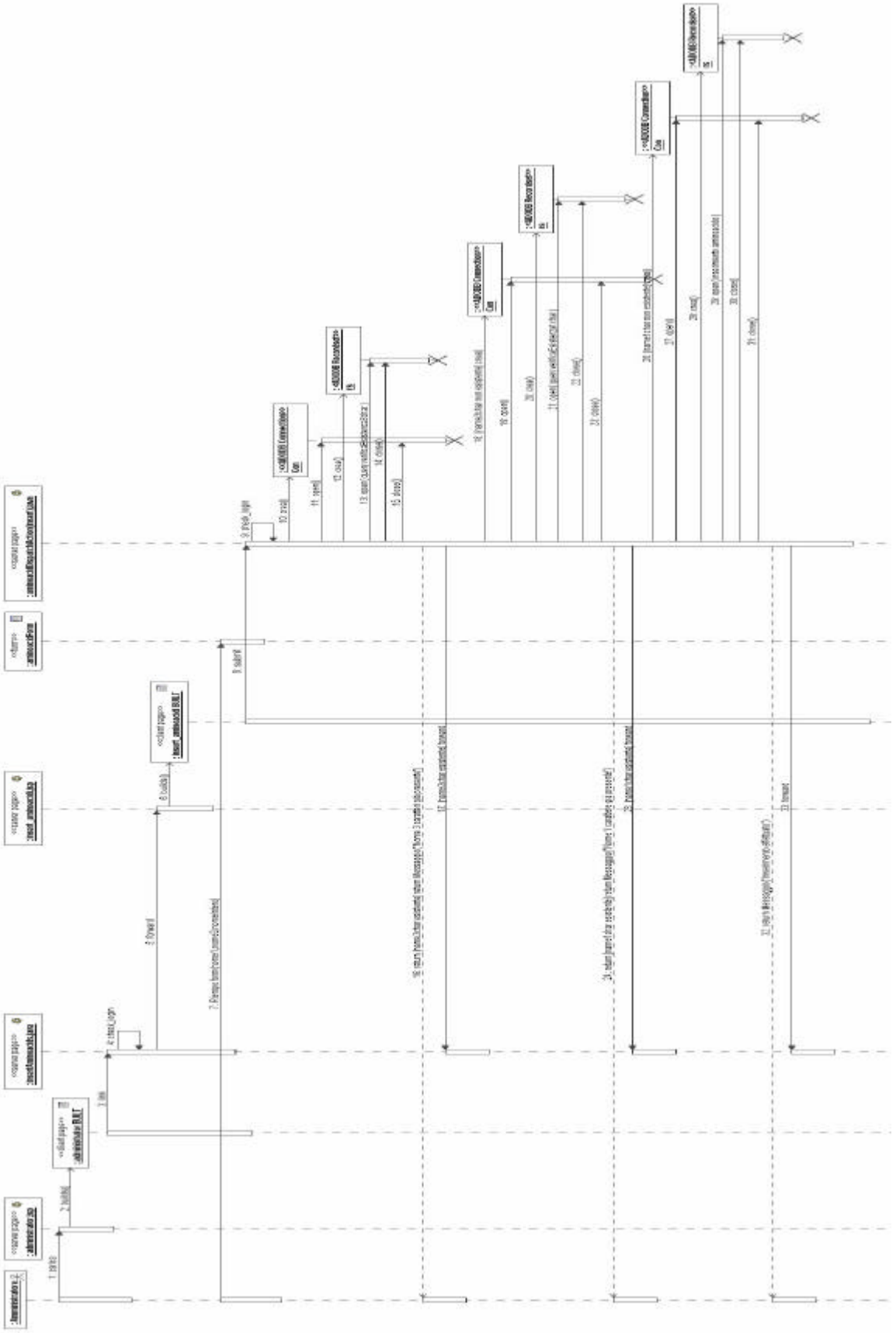
Caso d'uso: Visualizzazione codon e Modifica codon



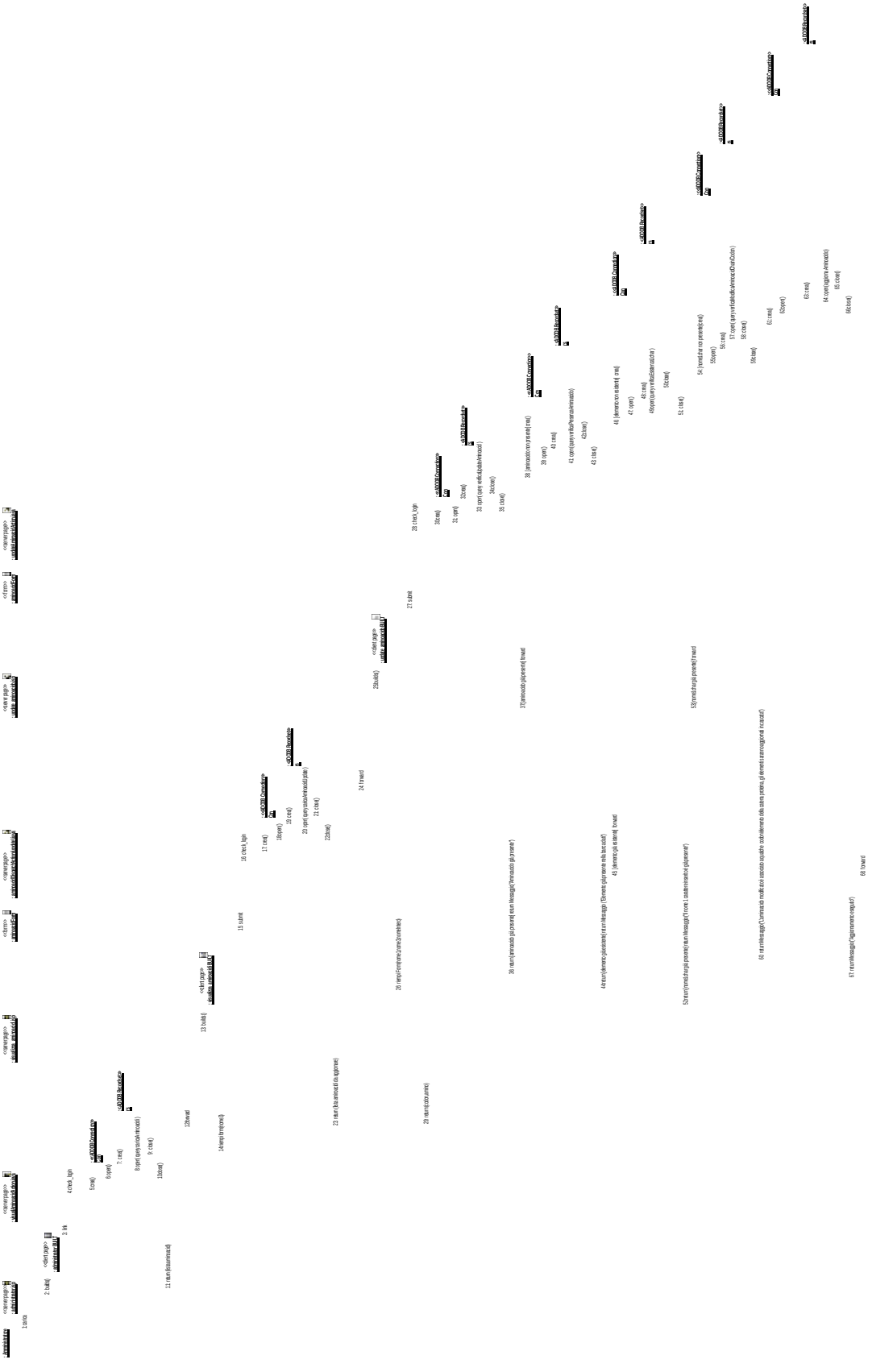
Caso d'uso: Visualizzazione codon ed Eliminazione codon



Caso d'uso: Inserimento aminoacido

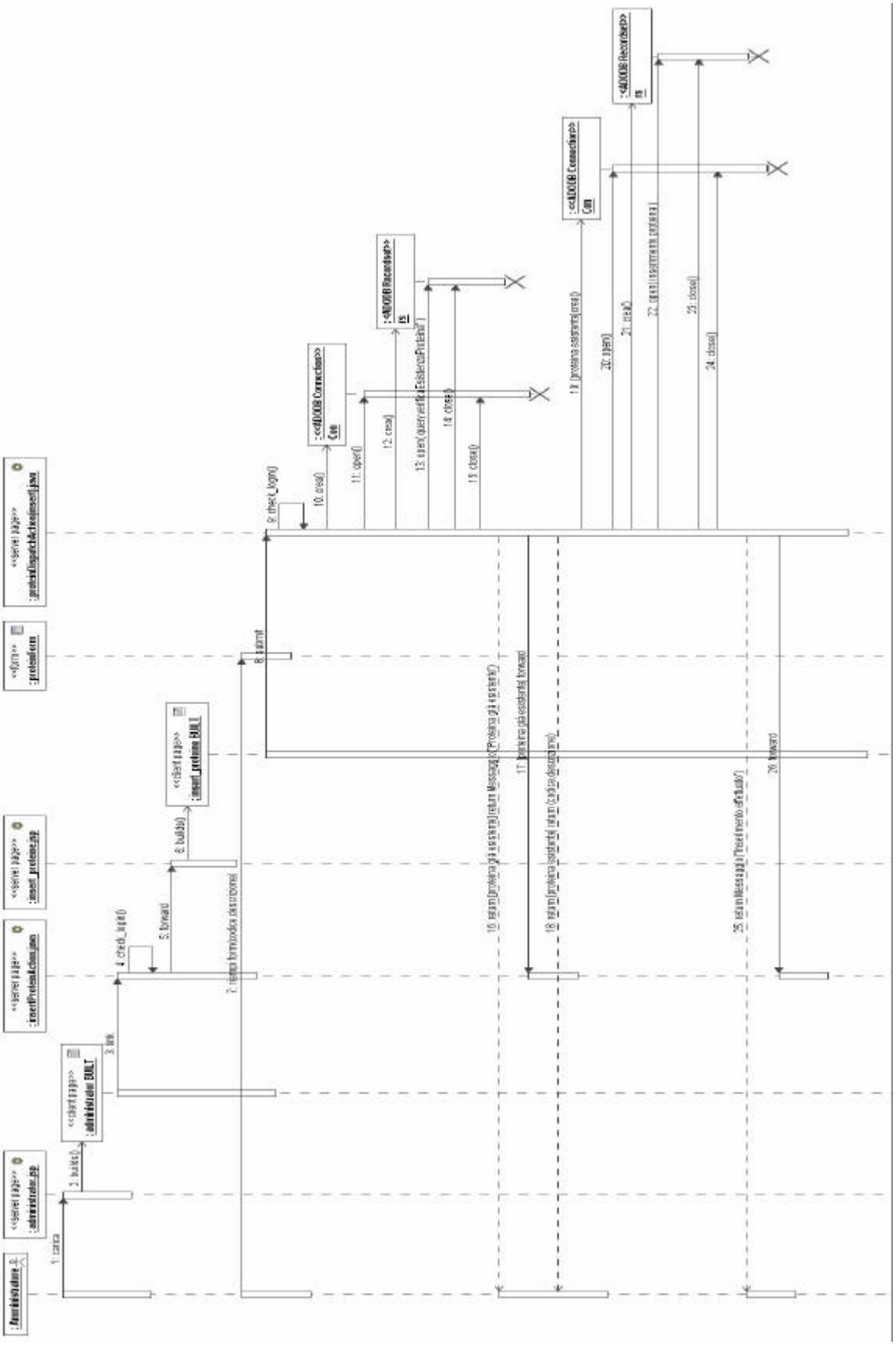


Caso d'uso: Visualizzazione aminoacido e Modifica aminoacido

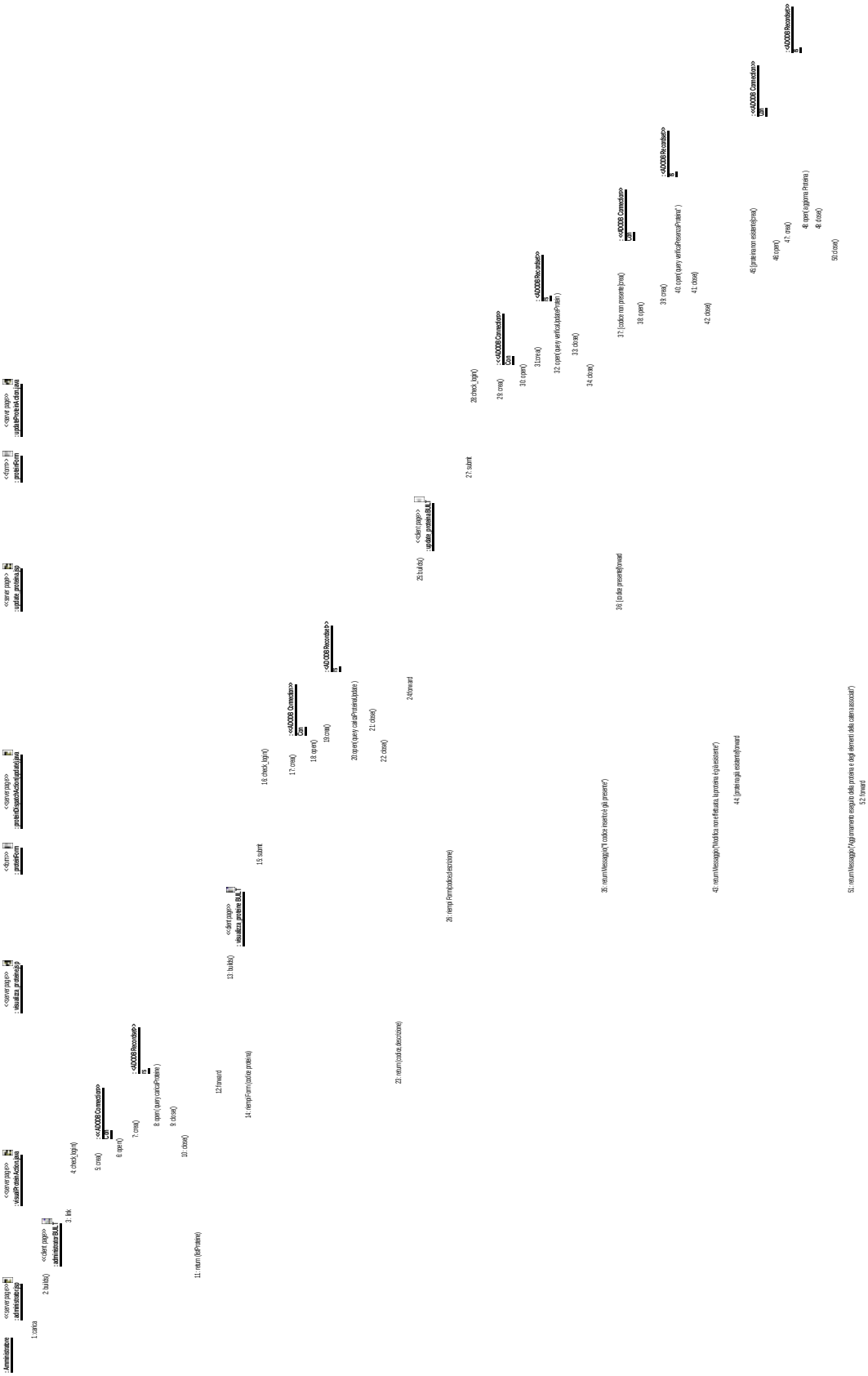


Caso d'uso: Visualizzazione aminoacido ed Eliminazione aminoacido

Caso d'uso: Inserimento Proteina

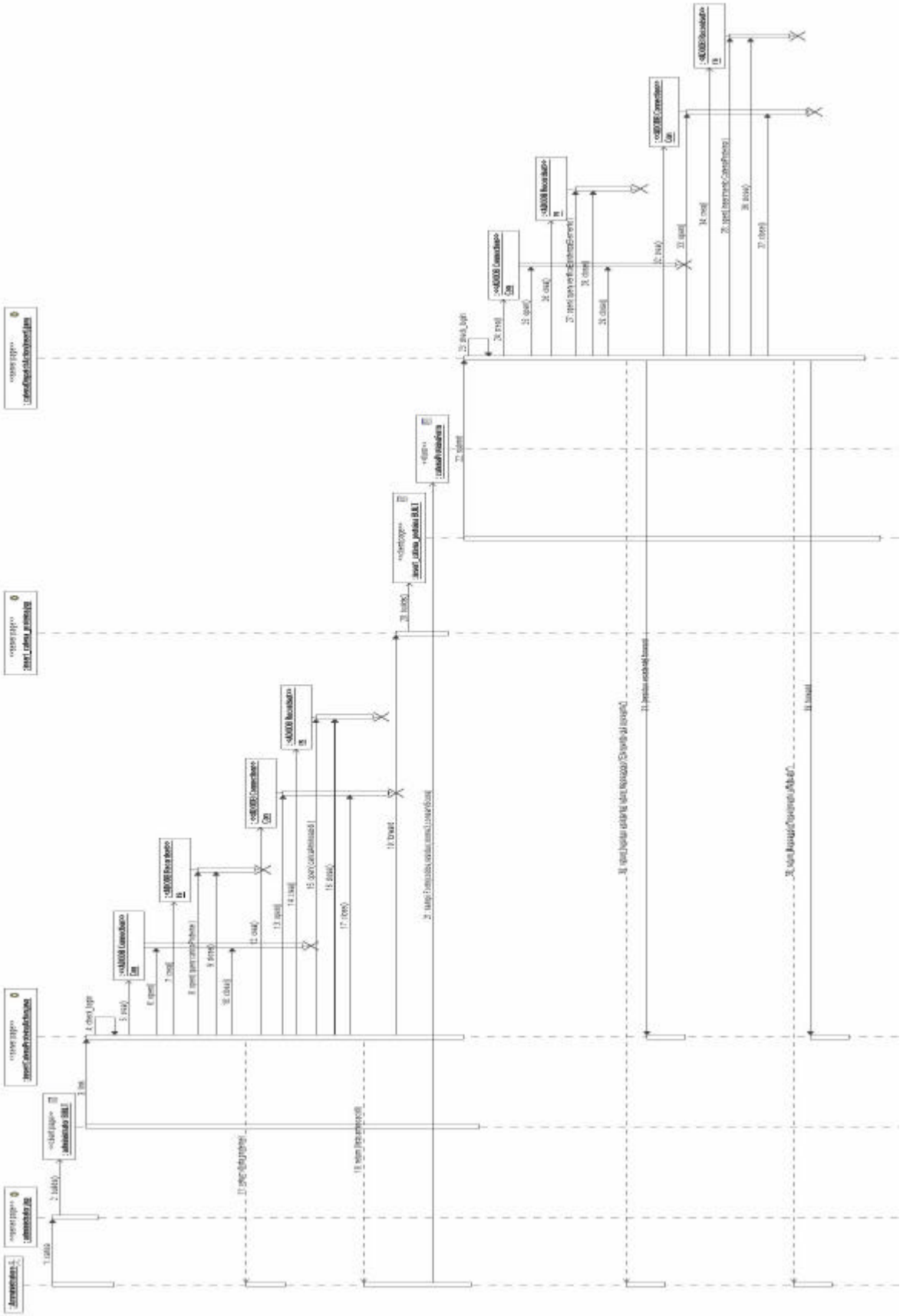


Caso d'uso: Visualizza proteine e Modifica proteine

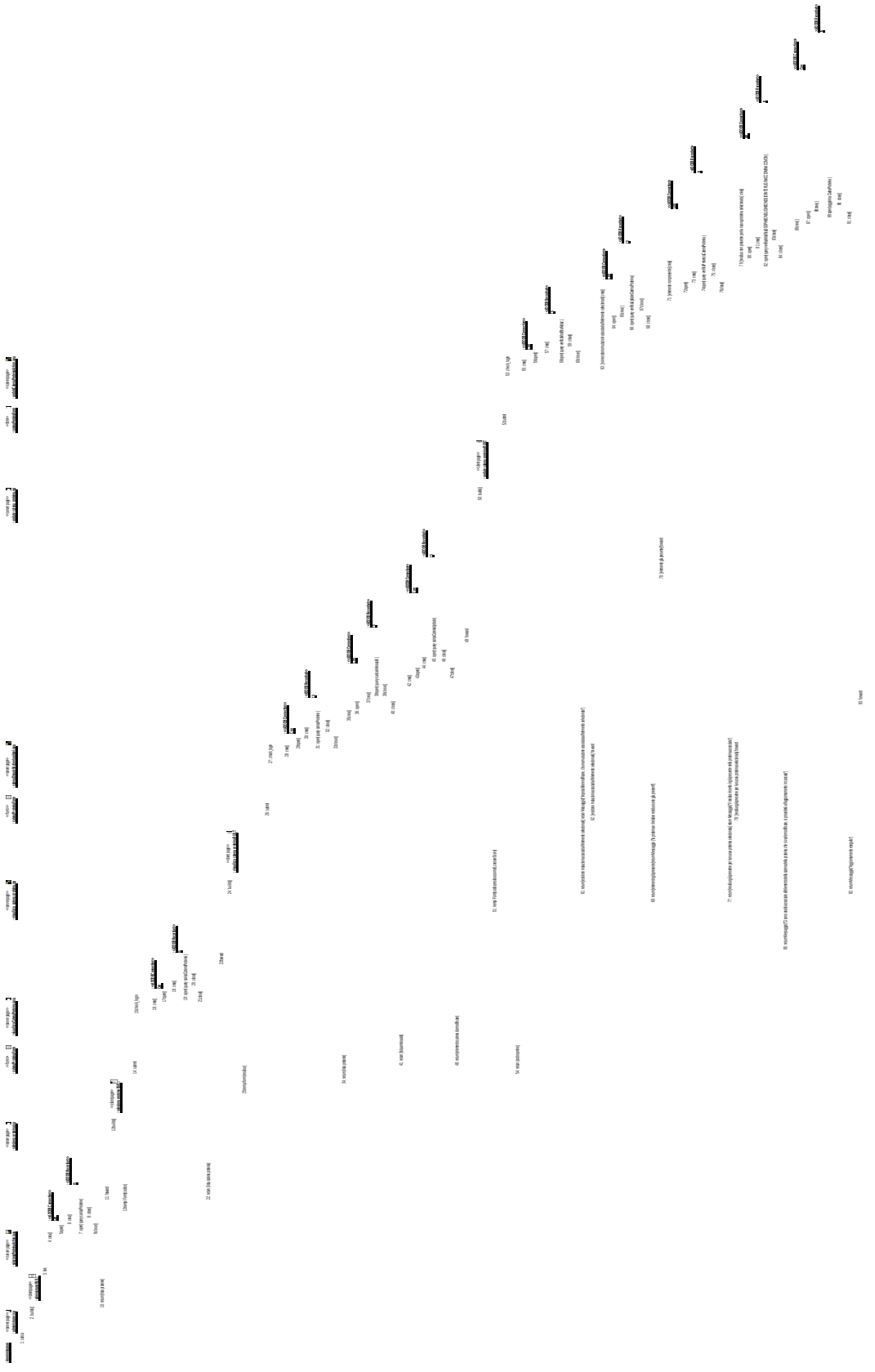


Caso d'uso: Visualizza proteine ed Elimina proteine

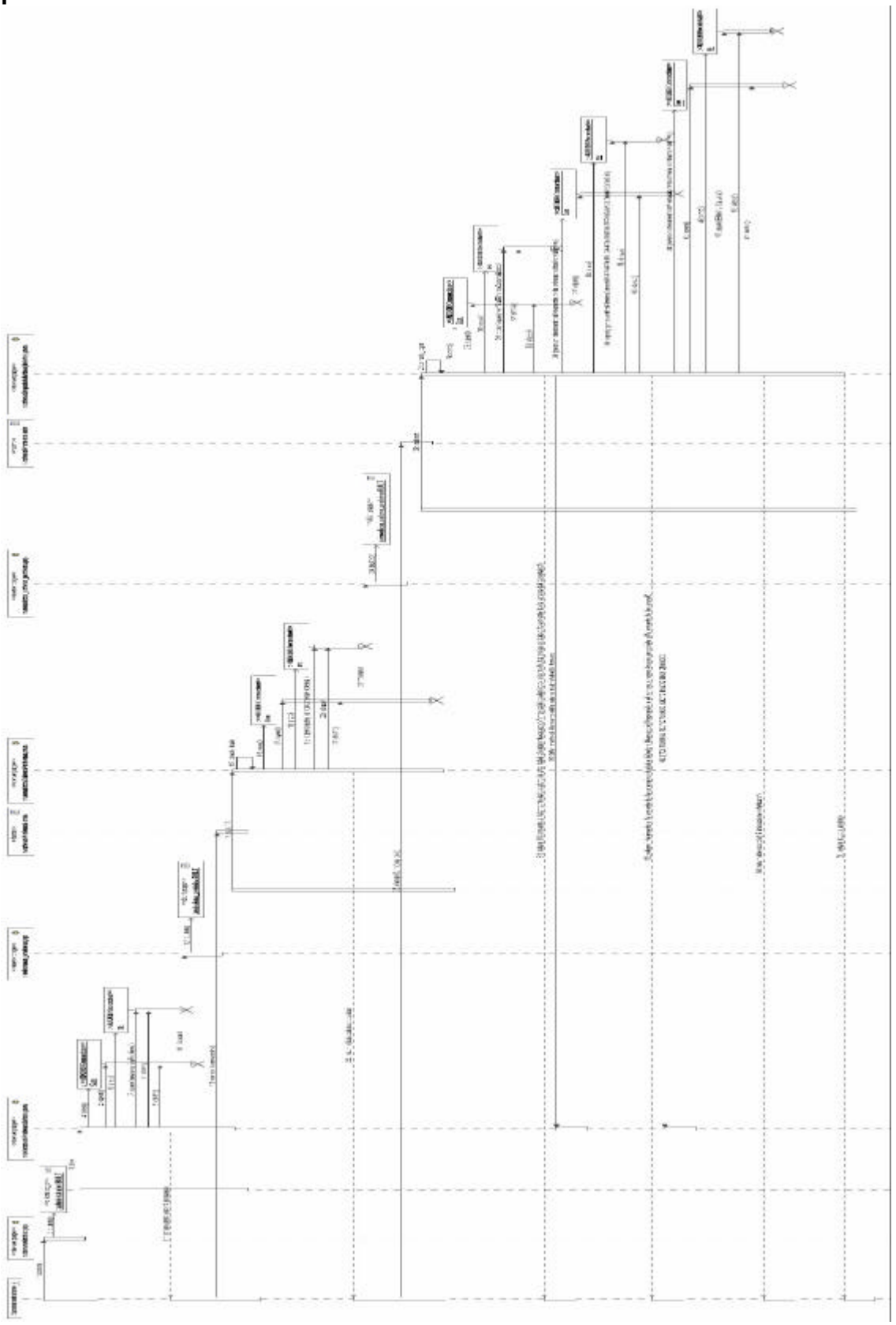
Caso d'uso: Inserimento Catena Proteina



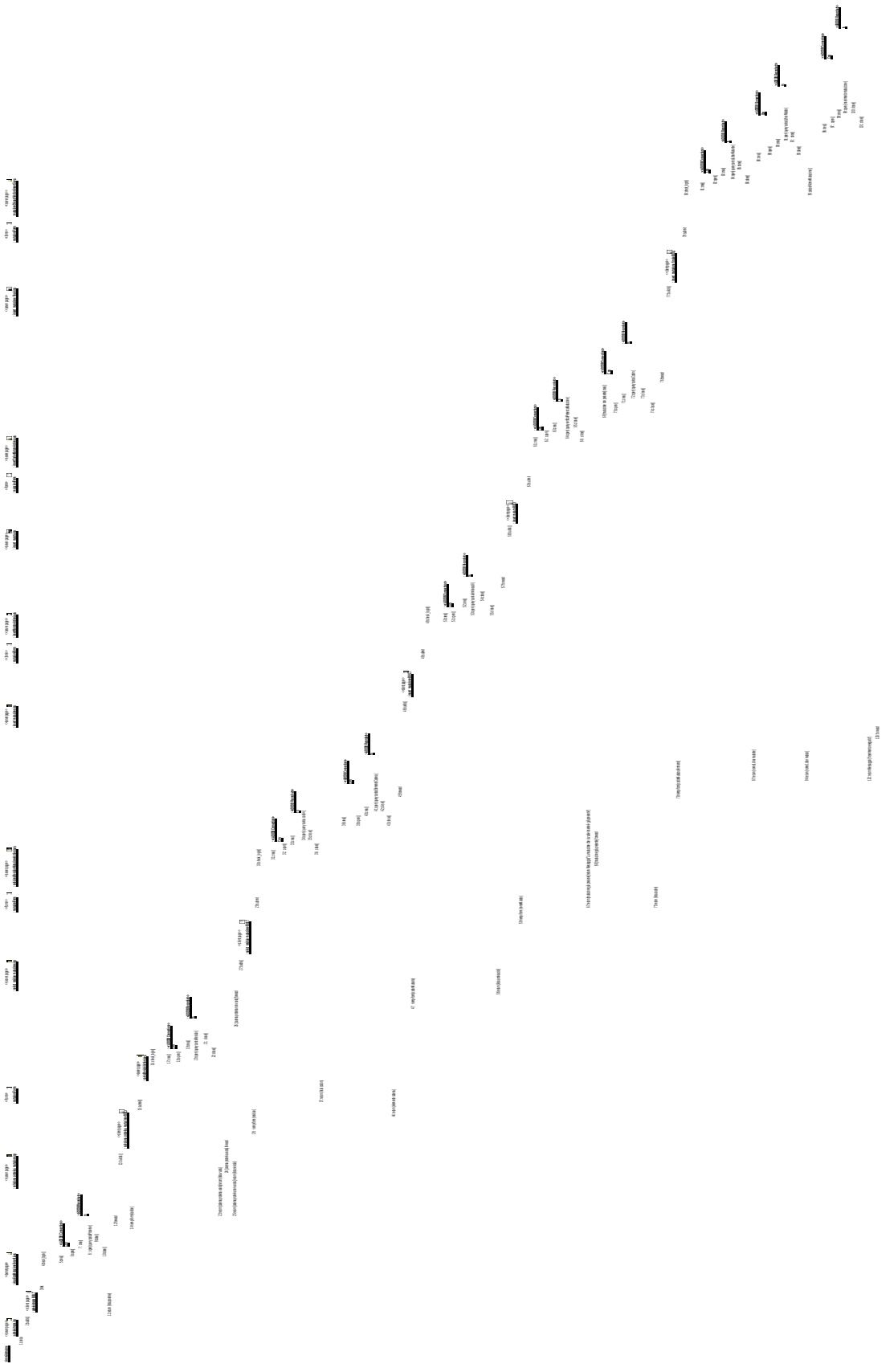
Caso d'uso: Visualizzazione Catena proteina e Modifica Catena proteina



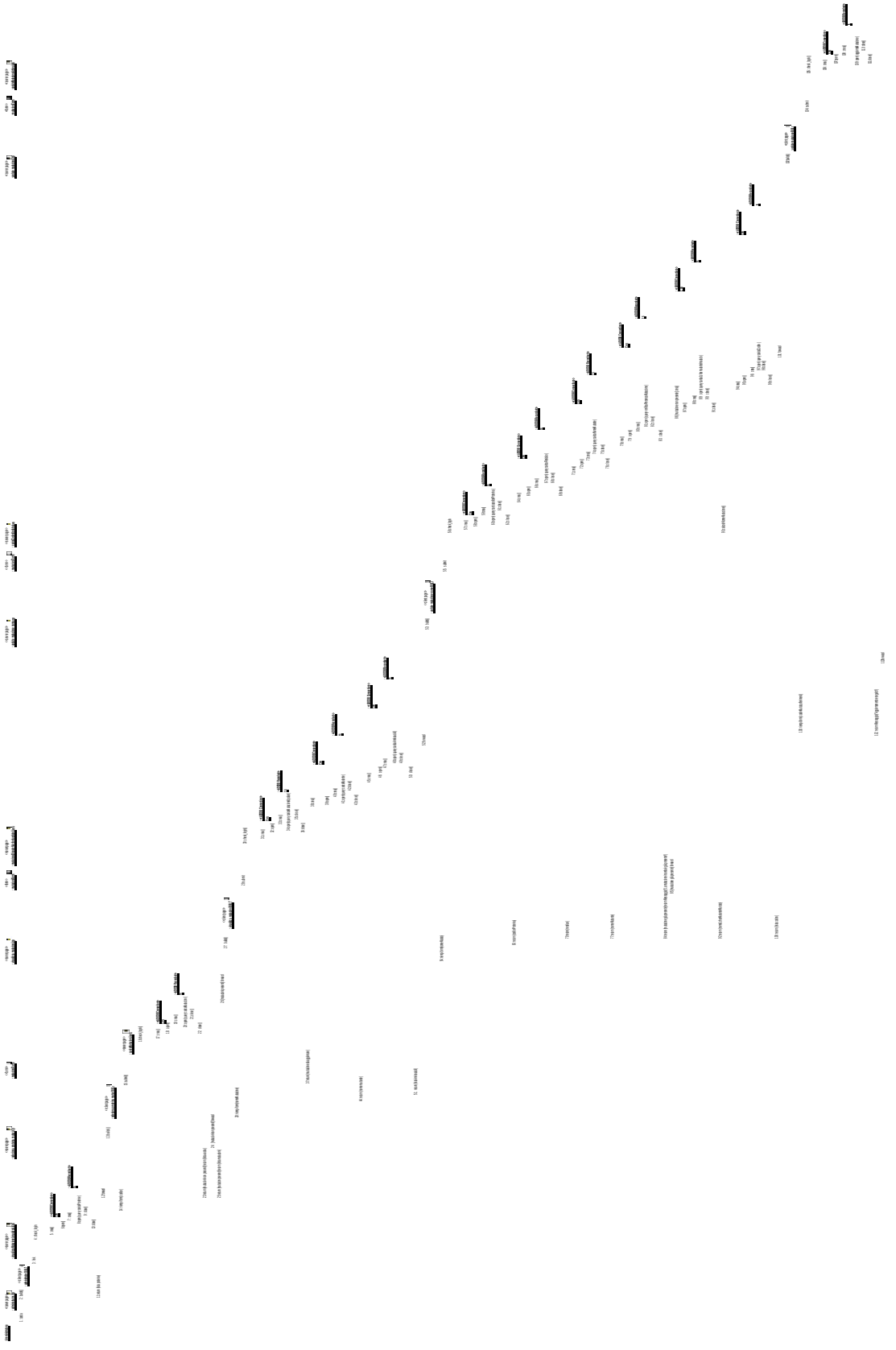
Caso d'uso: Visualizzazione Catena proteina ed Eliminazione Catena proteina



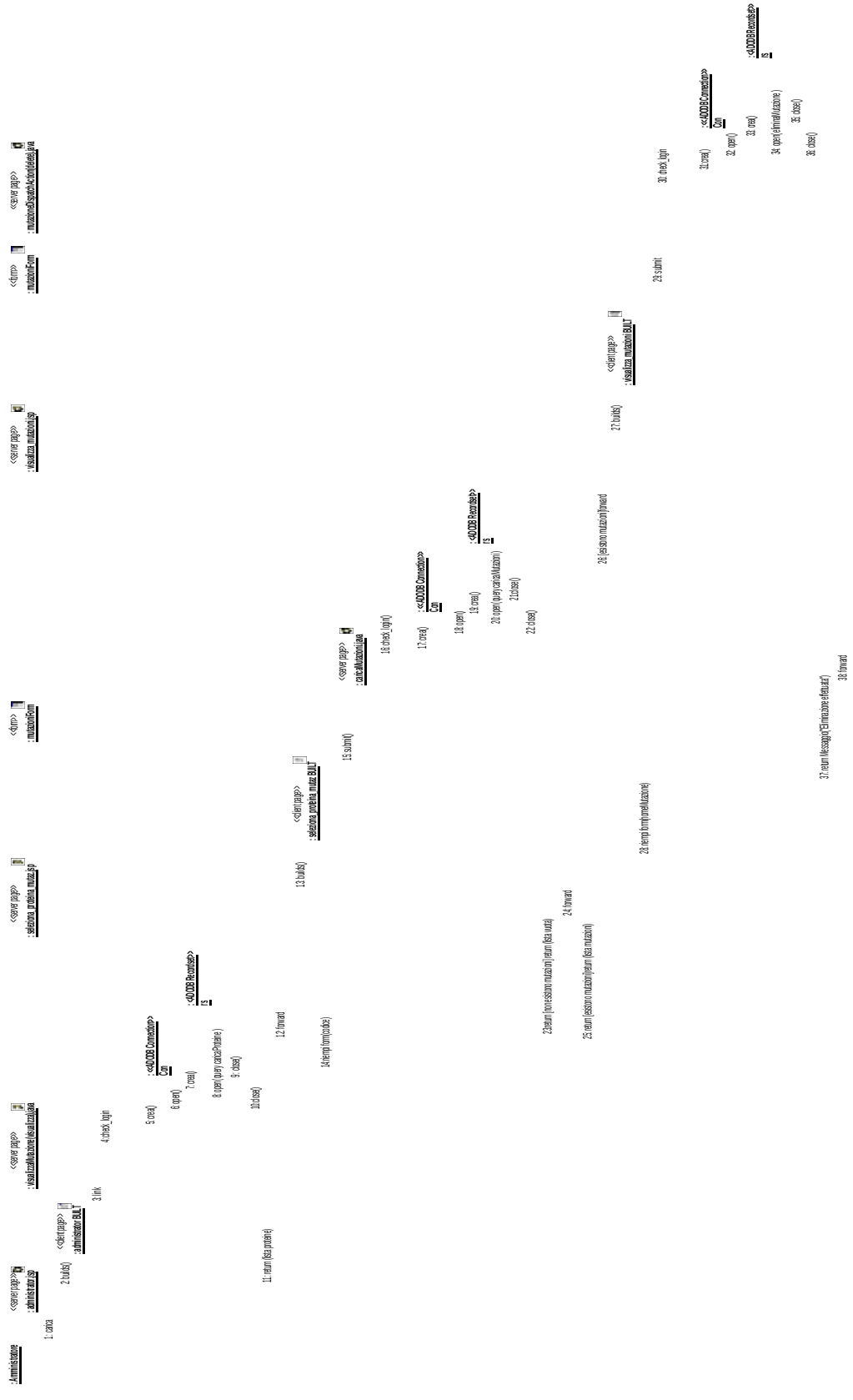
Caso d'uso: Inserimento mutazione missense



Caso d'uso: Visualizza Mutazione missense e Modifica Mutazione missense

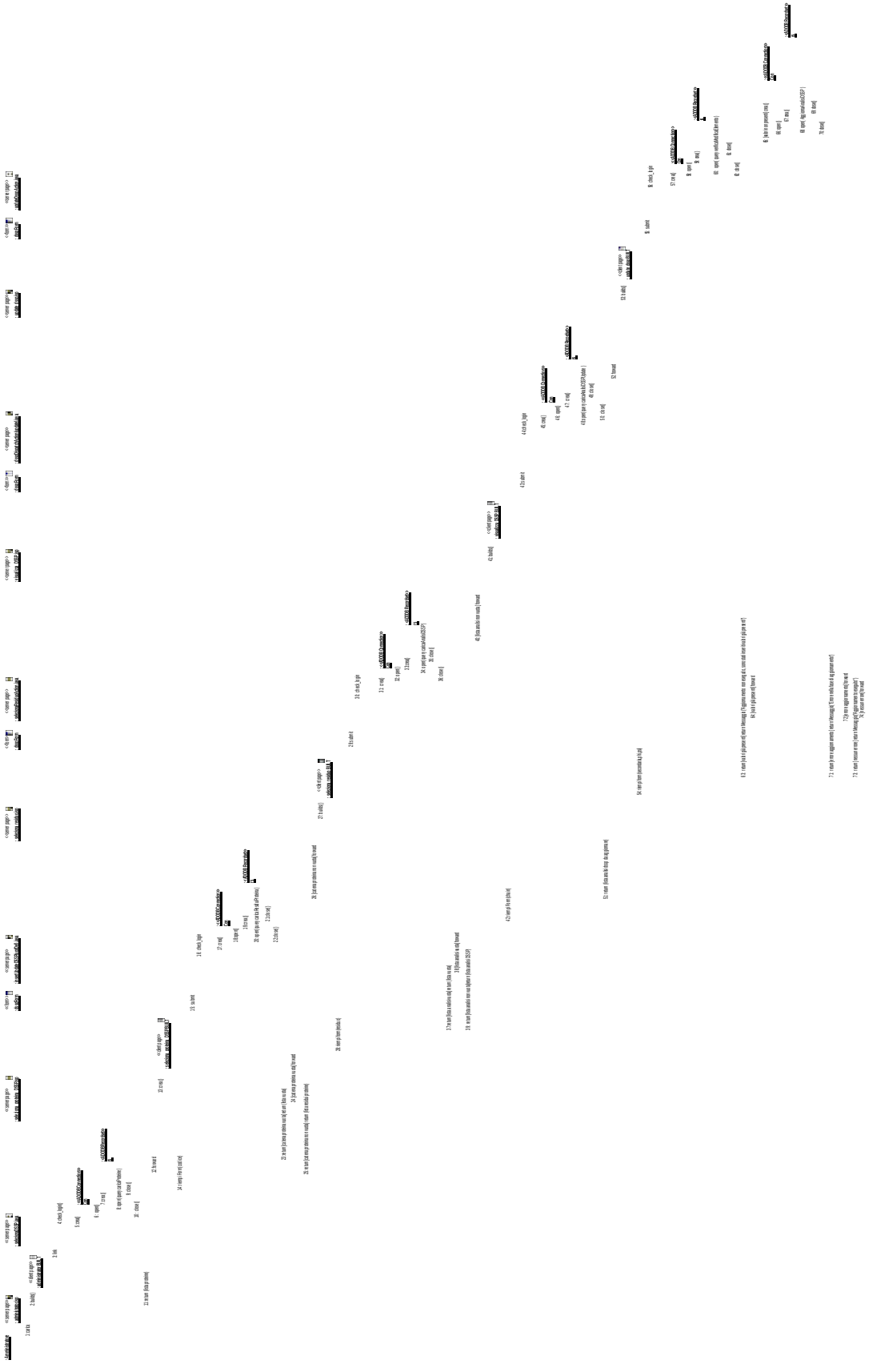


Caso d'uso: Visualizza Mutazione missense e Eliminazione Mutazione missense

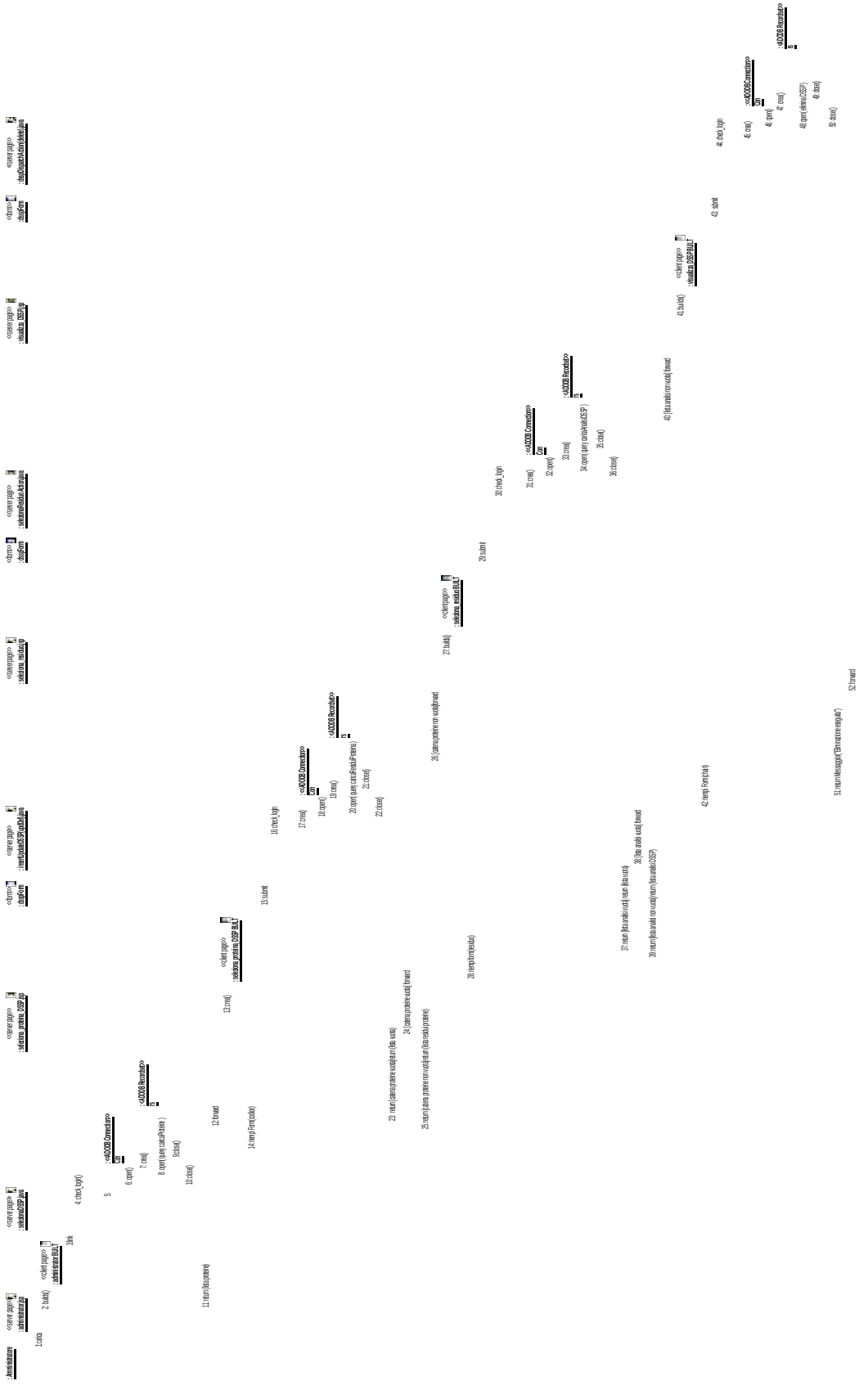


Caso d'uso: Inserimento Analisi DSSP

Caso d'uso: Visualizzazione Analisi DSSP e Modifica Analisi DSSP

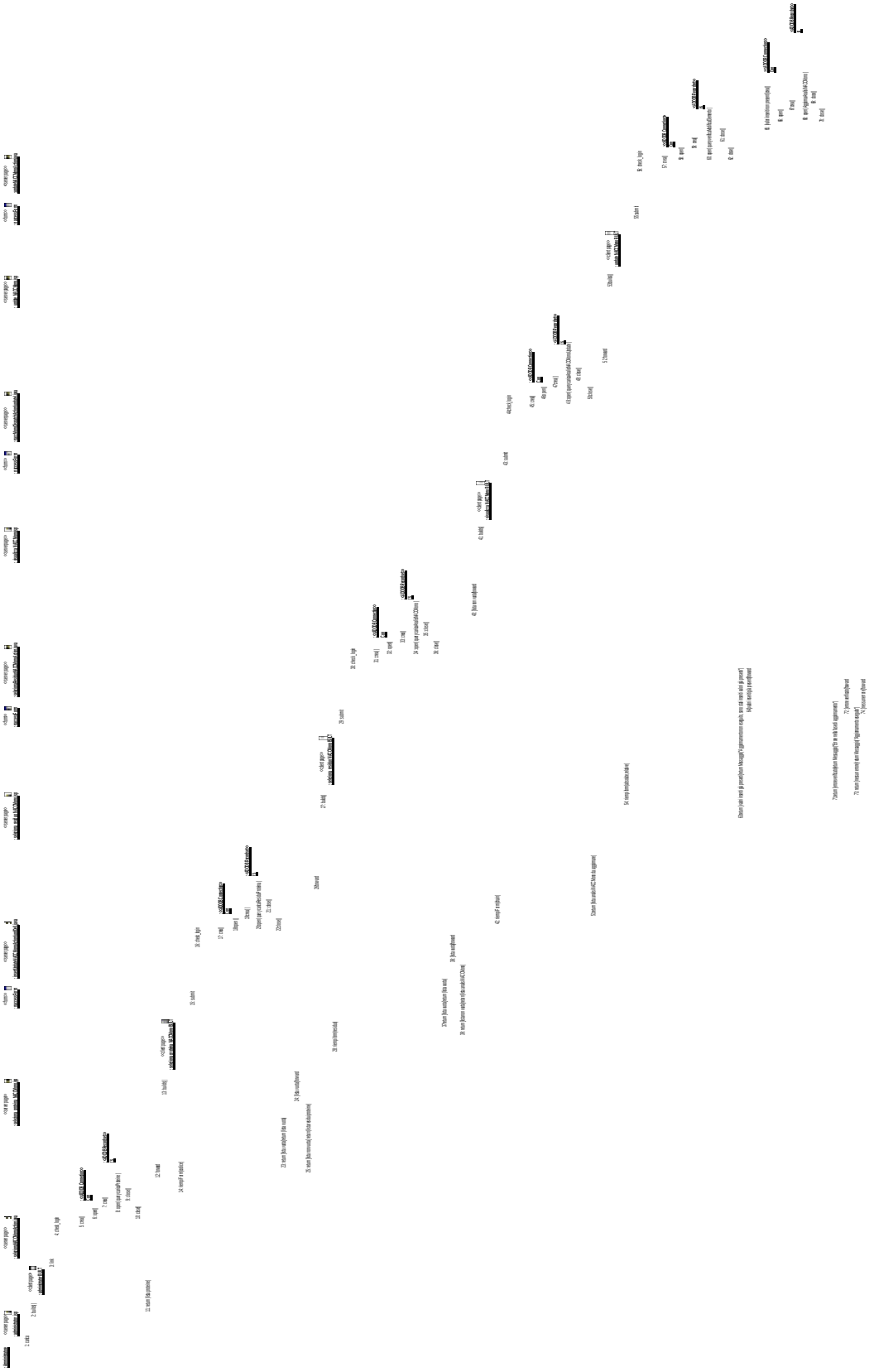


Caso d'uso: Visualizzazione Analisi DSSP e Eliminazione Analisi DSSP



Caso d'uso: Inserimento Analisi NACCESS Monomers

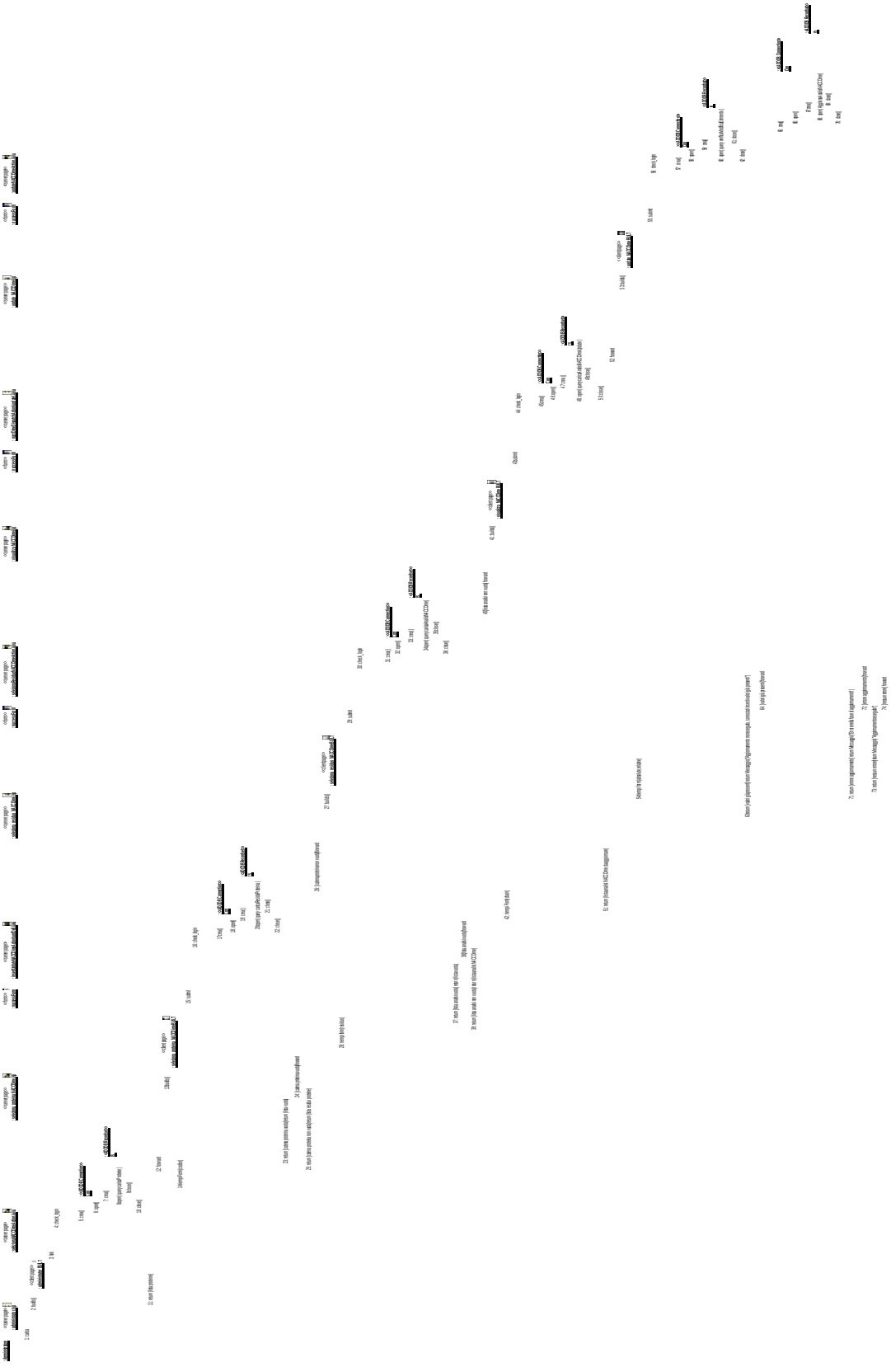
Caso d'uso: Visualizzazione Analisi NACCESS Monomers e Modifica Analisi NACCESS Monomers



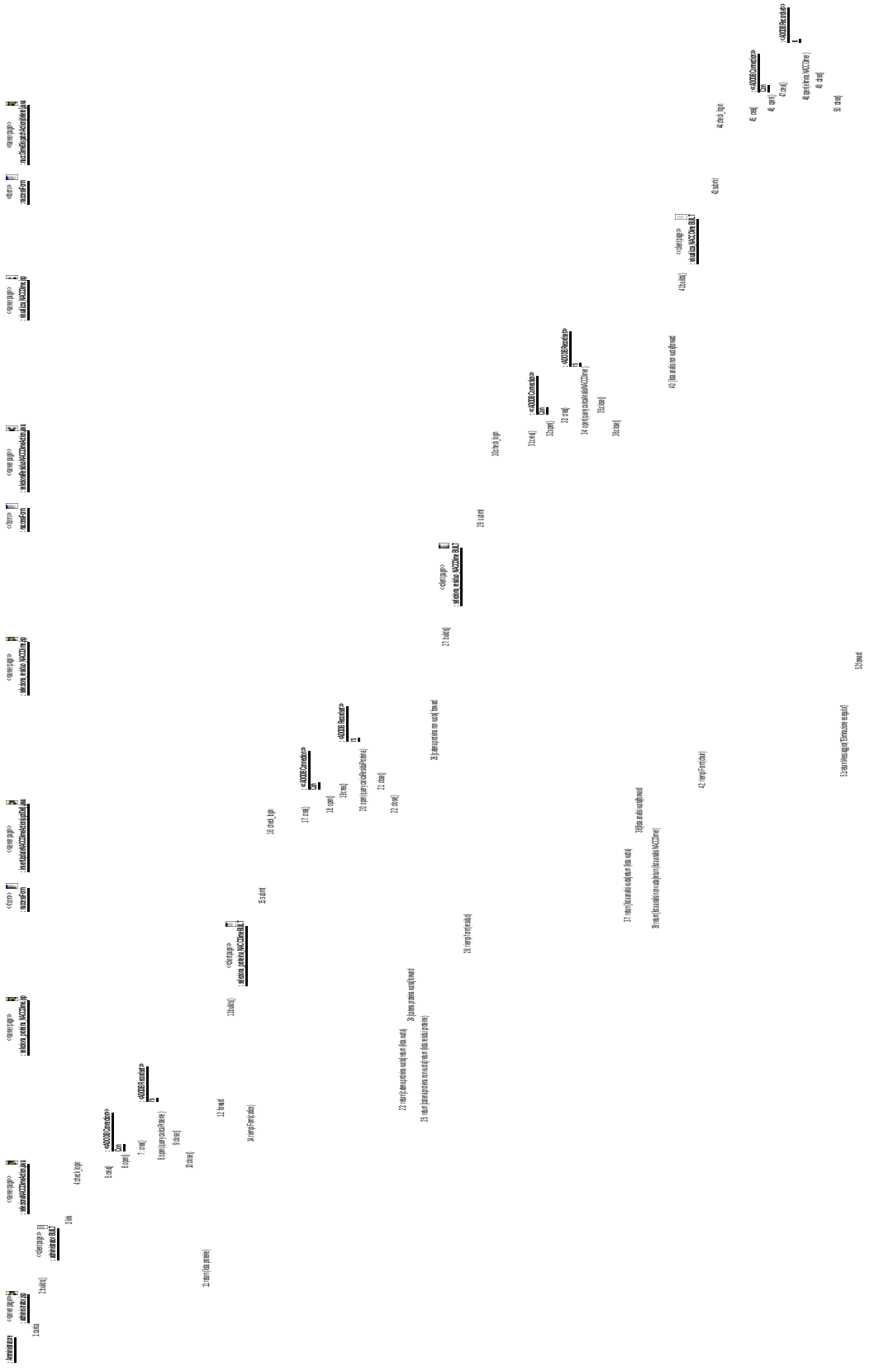
Caso d'uso: Visualizzazione Analisi NACCESS Monomers e Eliminazione Analisi NACCESS Monomers

Caso d'uso: Inserimento Analisi NACCESS Dimer

Caso d'uso: Visualizzazione Analisi NACCESS Dimer e Modifica Analisi NACCESS Dimer

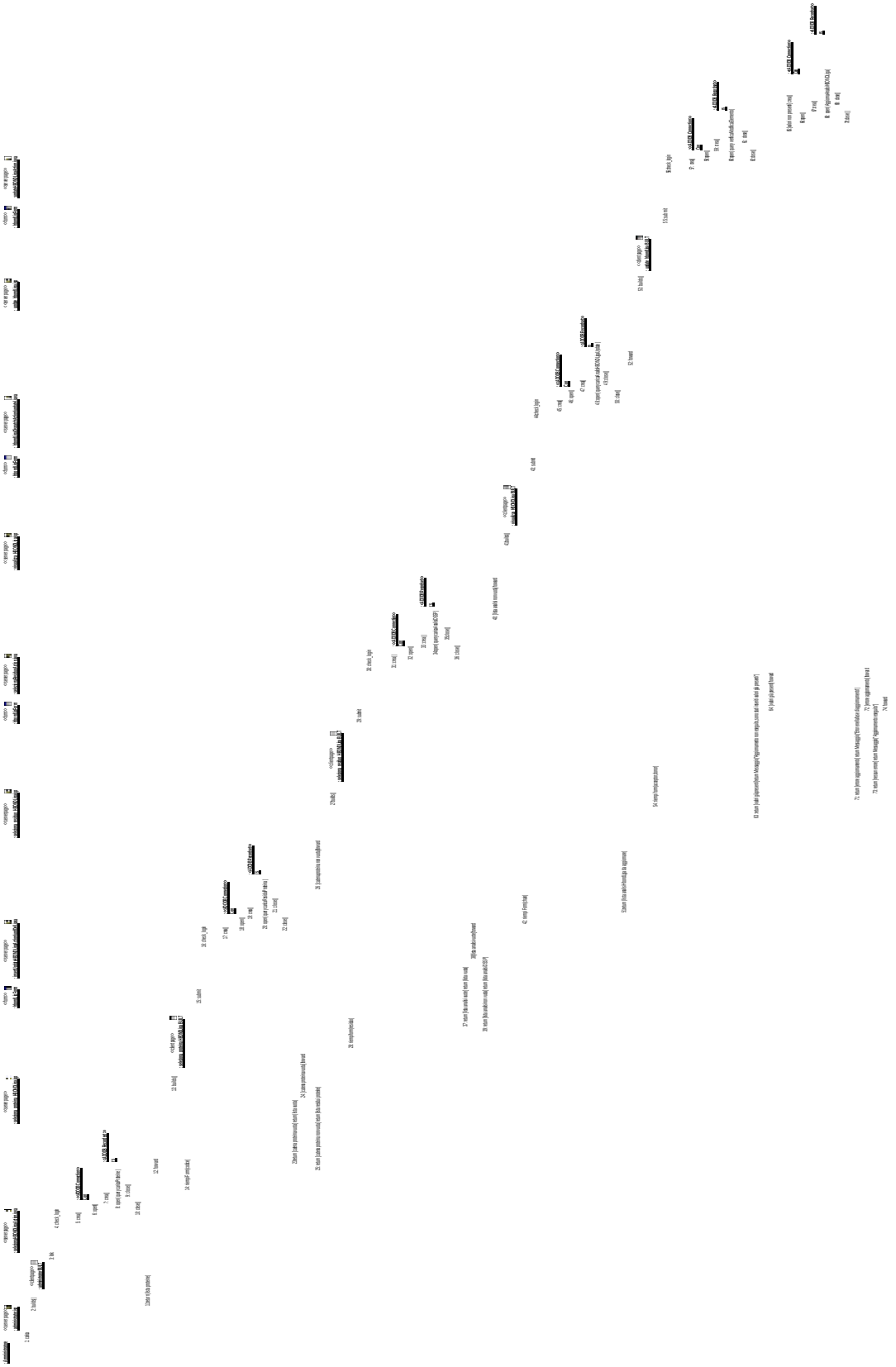


Caso d'uso: Visualizzazione Analisi NACCESS Dimer e Eliminazione Analisi NACCESS Dimer

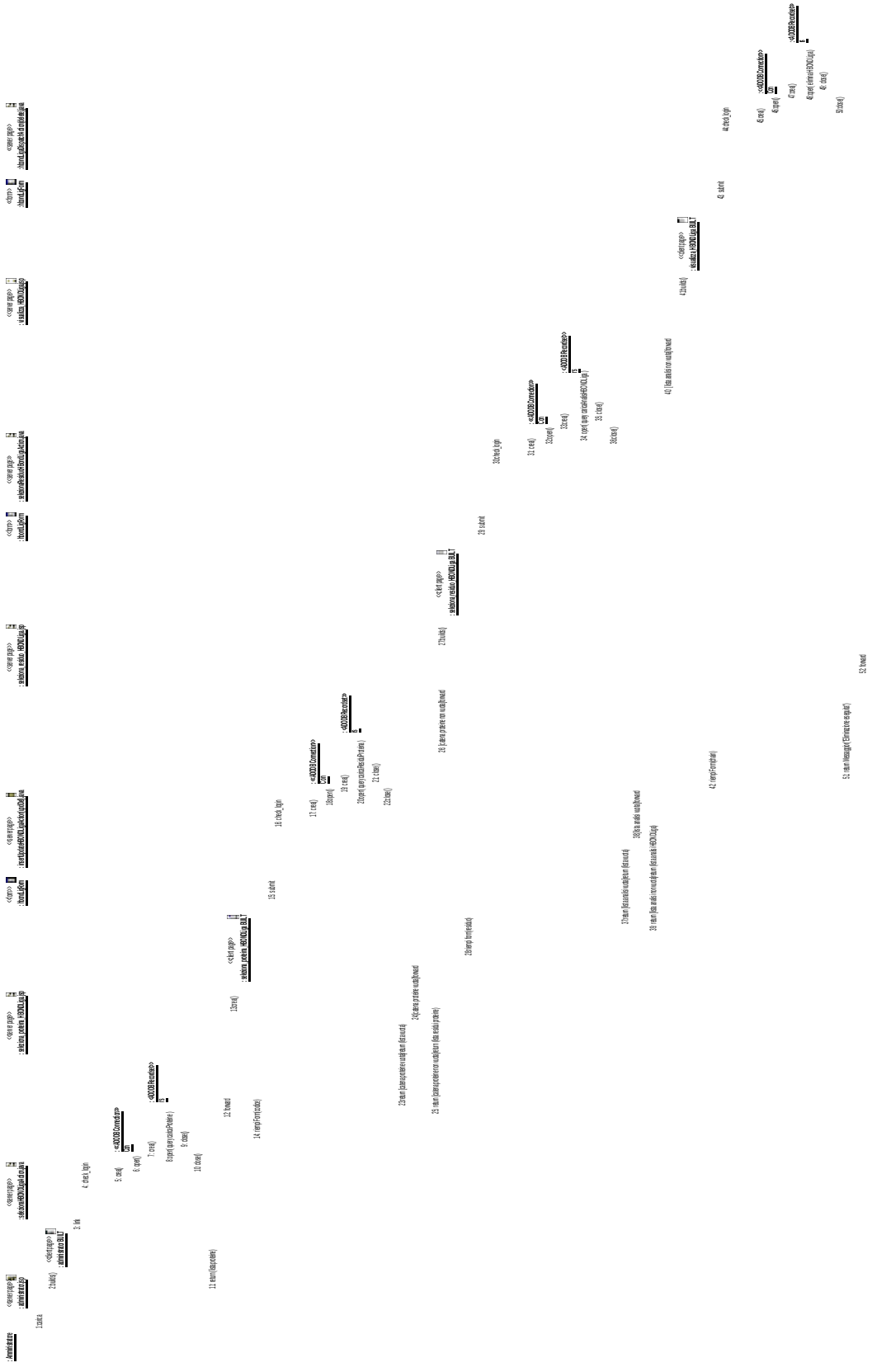


Caso d'uso: Inserimento Analisi H-BOND con Ligando

Caso d'uso: Visualizzazione Analisi H-BOND con Ligando e Modifica Analisi H-BOND con Ligando

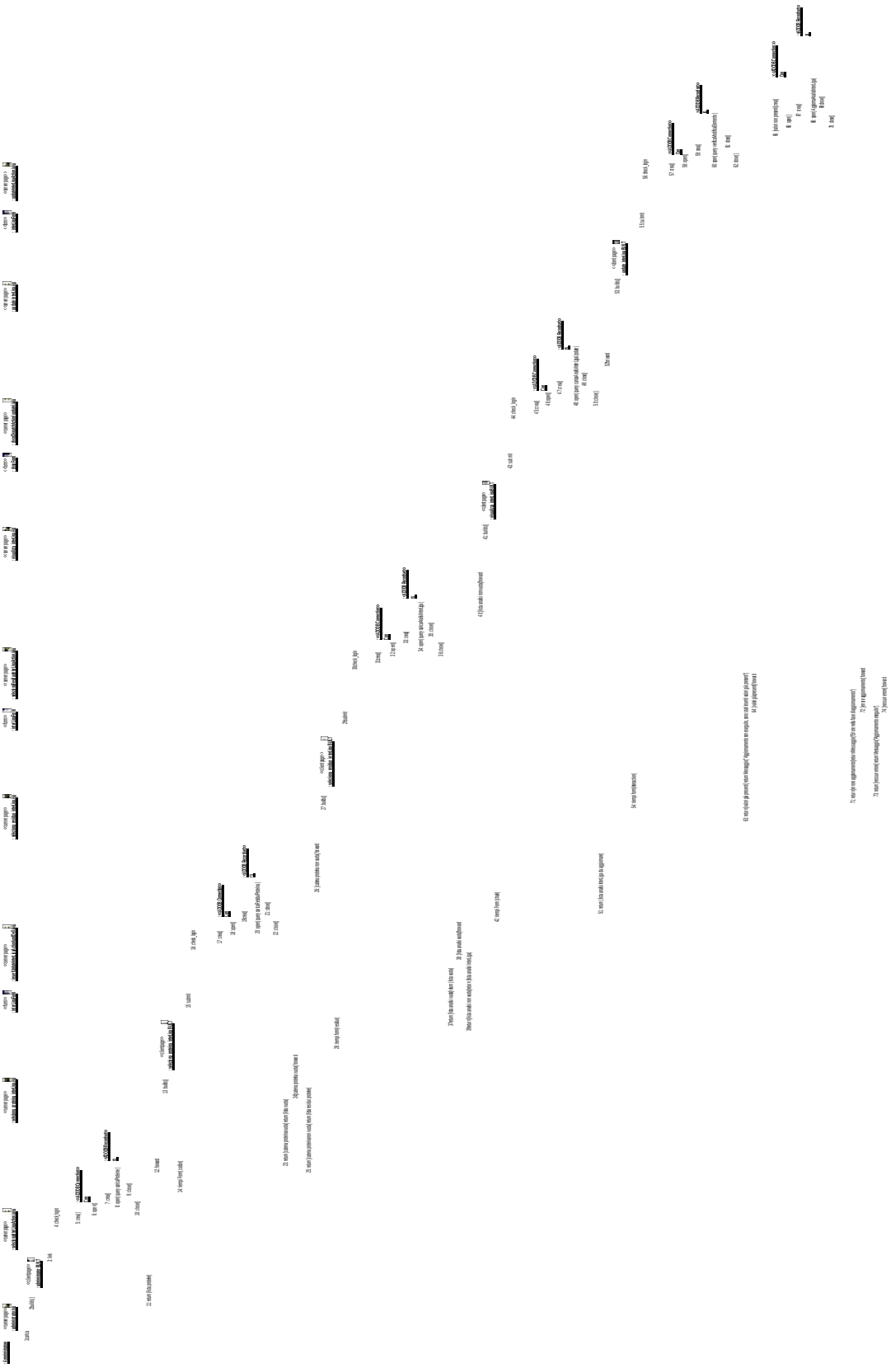


Caso d'uso: Visualizzazione Analisi H-BOND con Ligando e Eliminazione Analisi H-BOND con Ligando

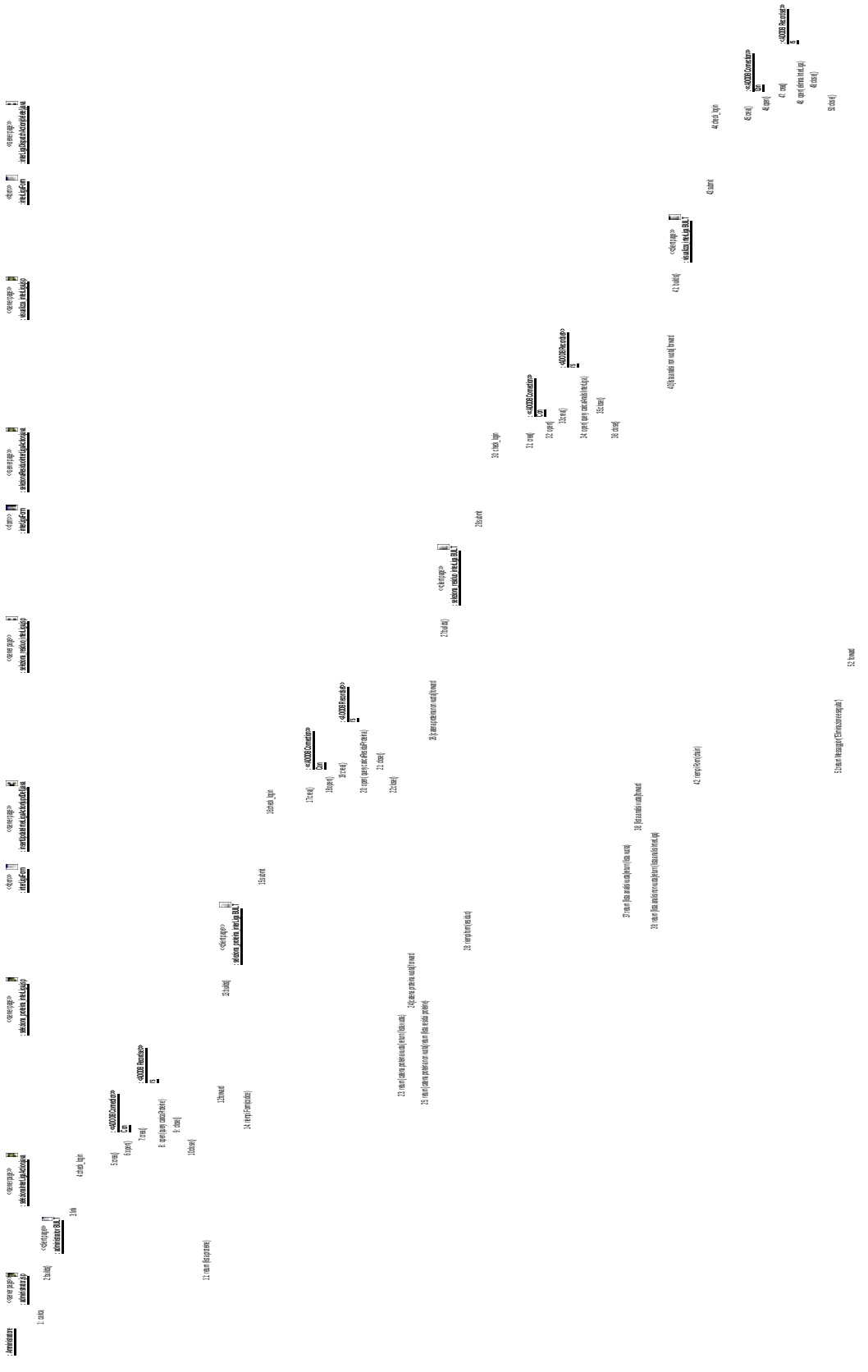


Caso d'uso: Inserimento Analisi Interazioni con Ligando

Caso d'uso: Visualizzazione Analisi Interazioni con Ligando e Modifica Analisi Interazioni con Ligando

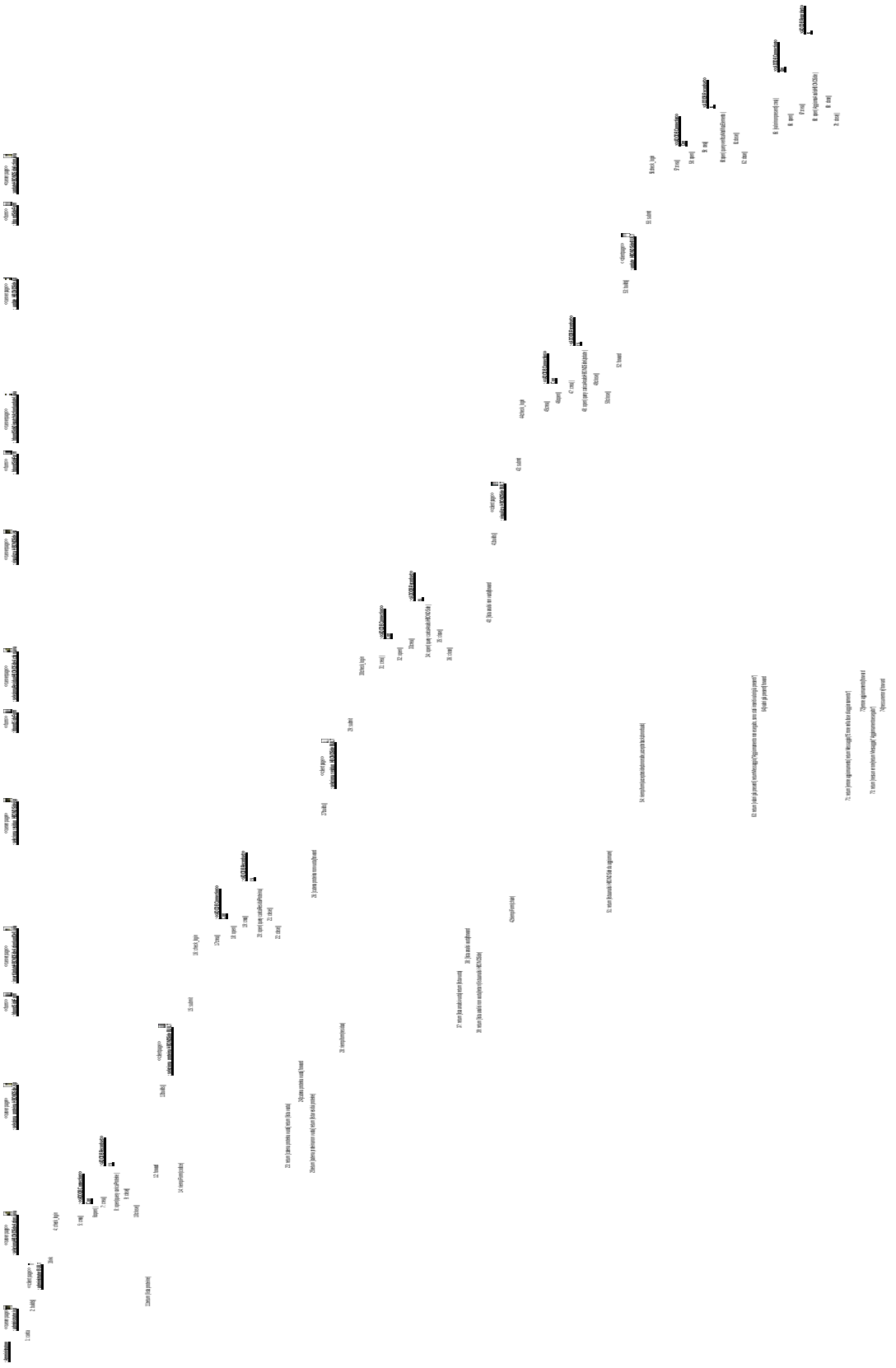


Caso d'uso: Visualizzazione Analisi Interazioni con Ligando e Eliminazione Analisi Interazioni con Ligando

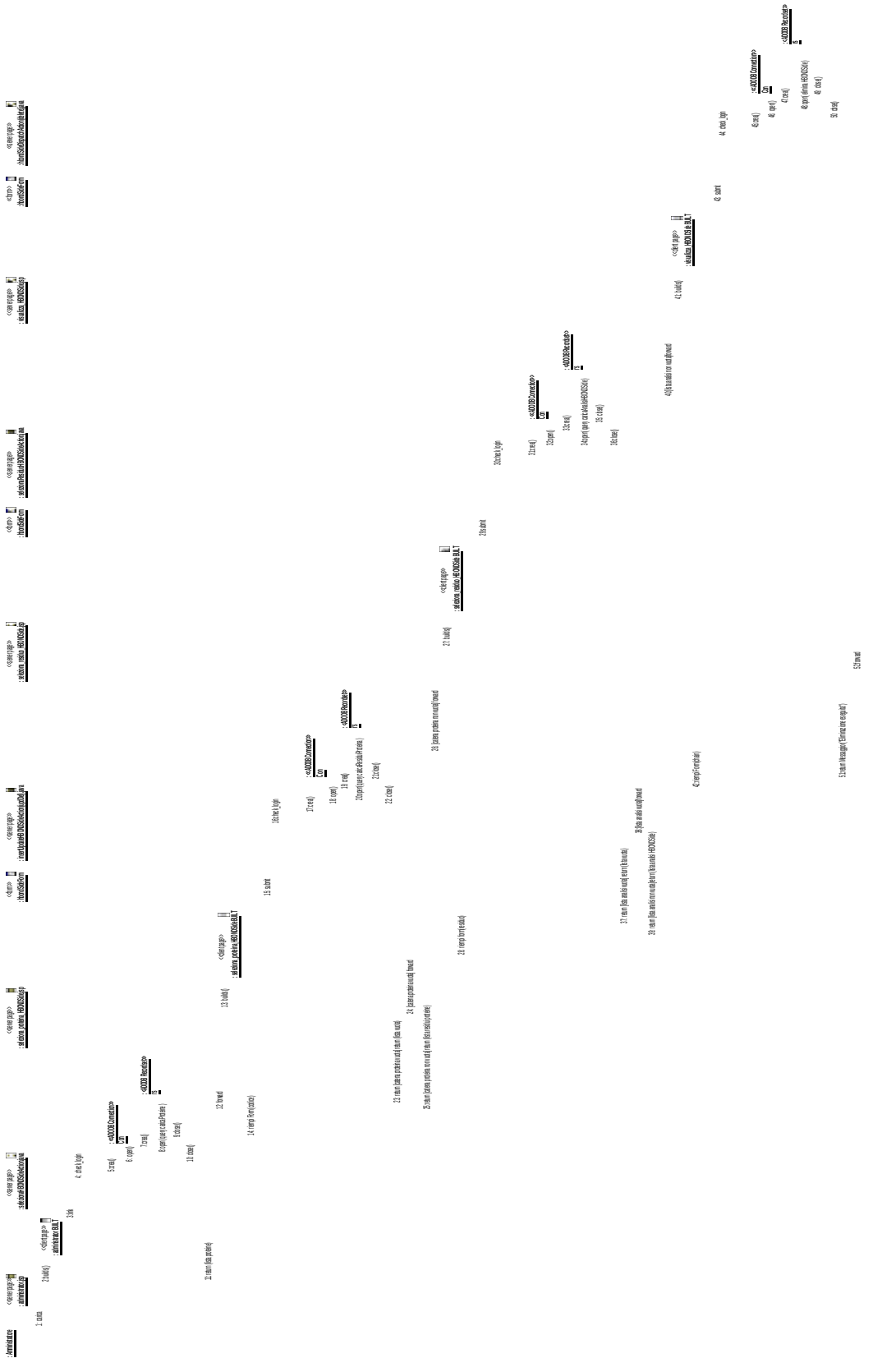


Caso d’uso: Inserimento Analisi H-BOND con Sidechains

Caso d'uso: Visualizzazione Analisi H-BOND con SideChains e Modifica Analisi H-BOND con SideChains

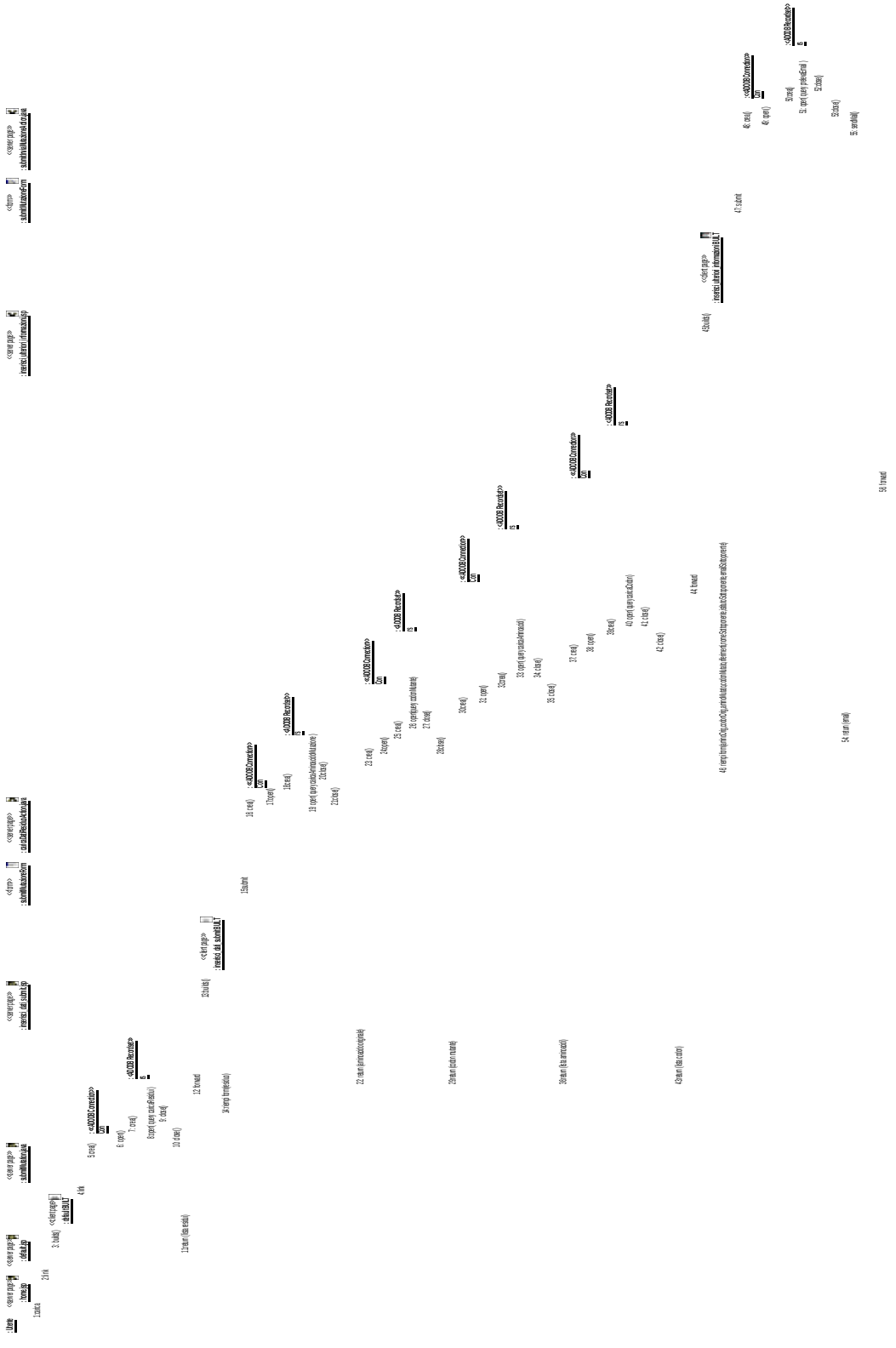


Caso d'uso: Visualizzazione Analisi H-BOND con SideChains e Eliminazione Analisi H-BOND con SideChains

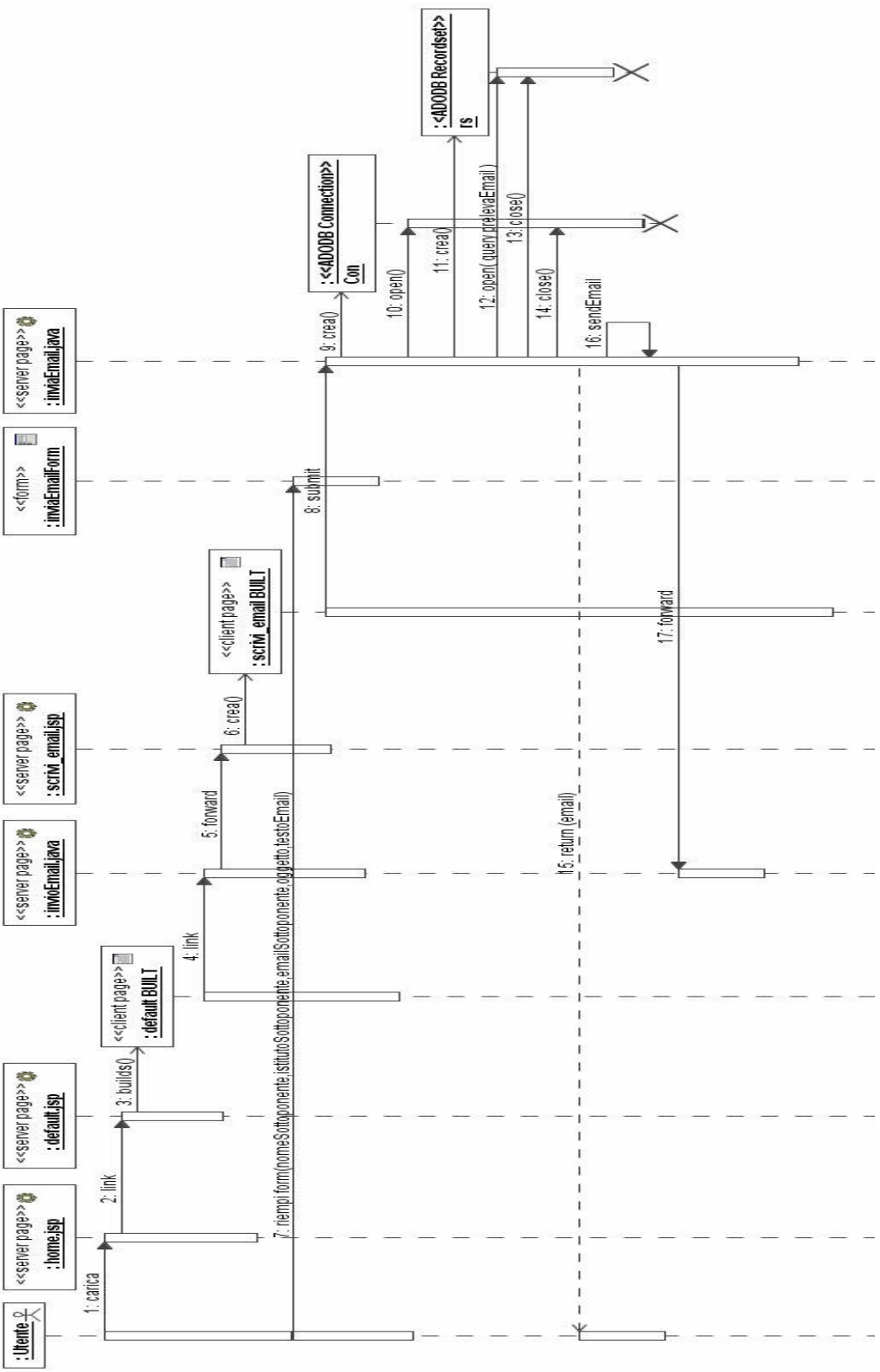


❖ Modulo Pubblico

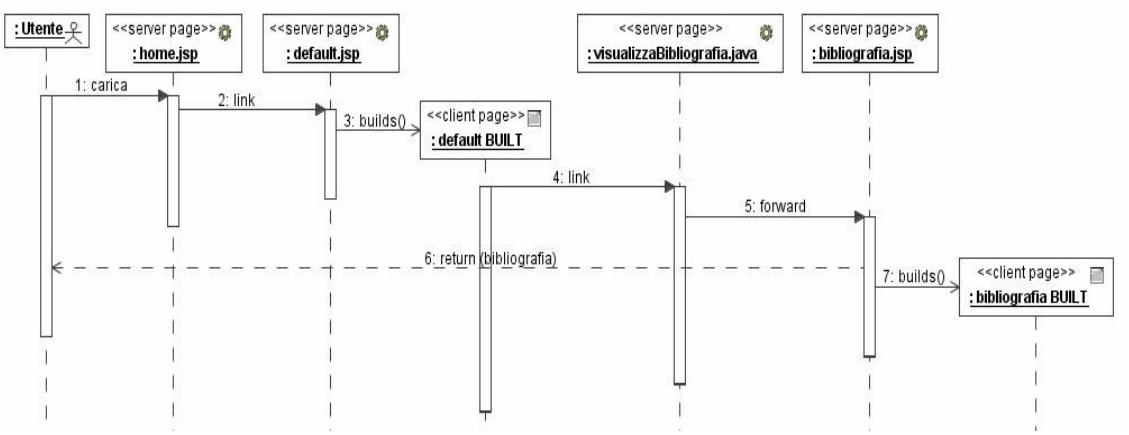
Caso d'uso: Submit nuova mutazione



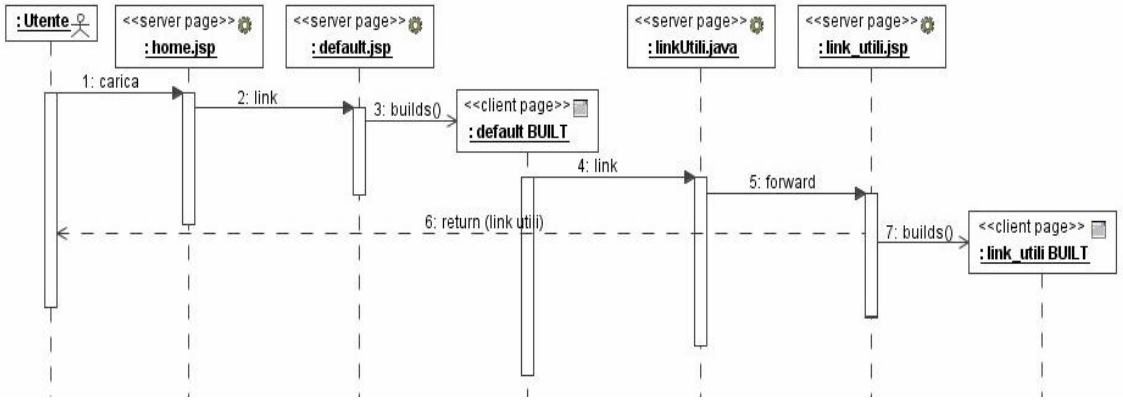
Caso d'uso: Invia e-mail all'amministratore



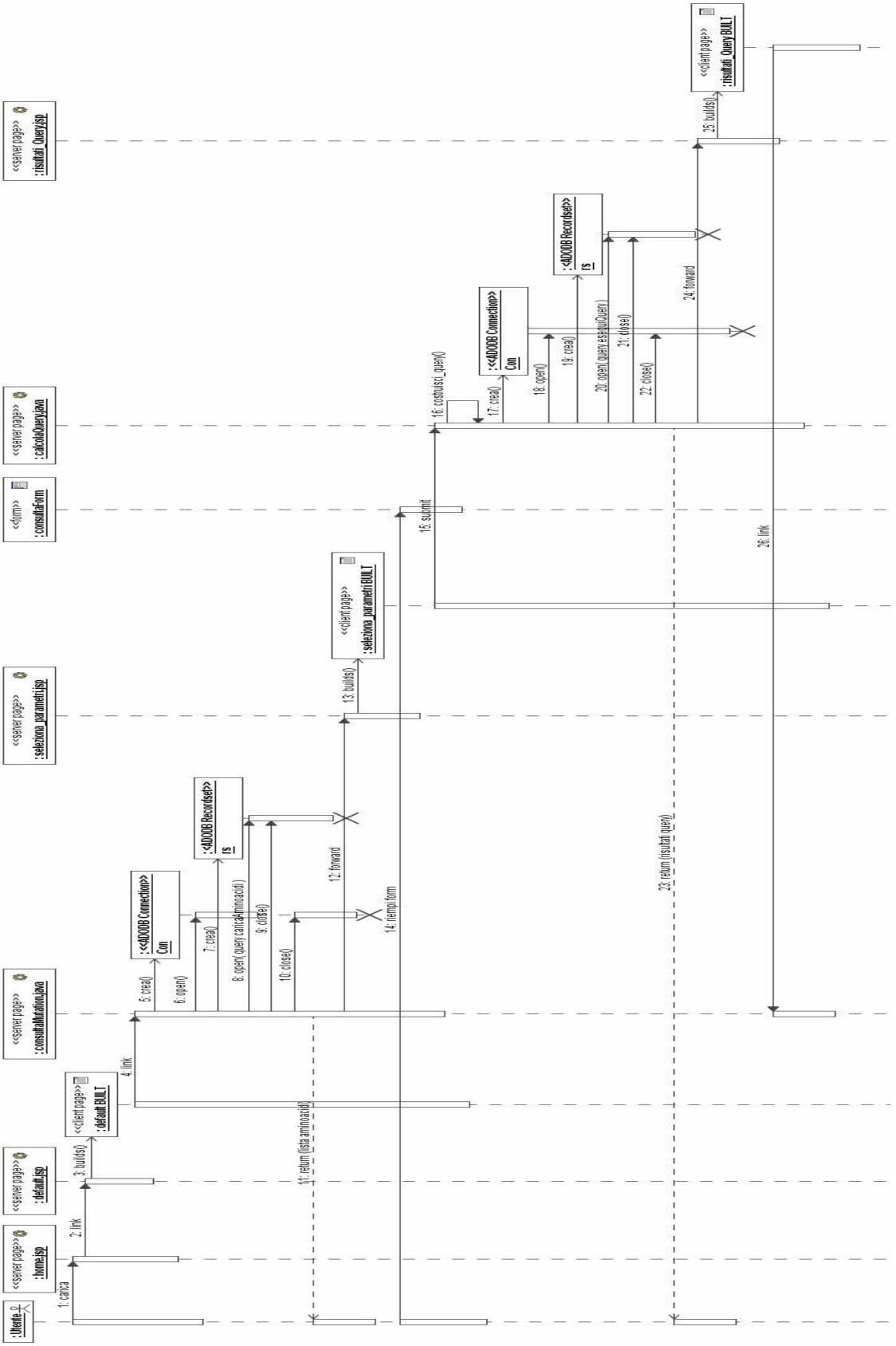
Caso d'uso: Visualizza bibliografia



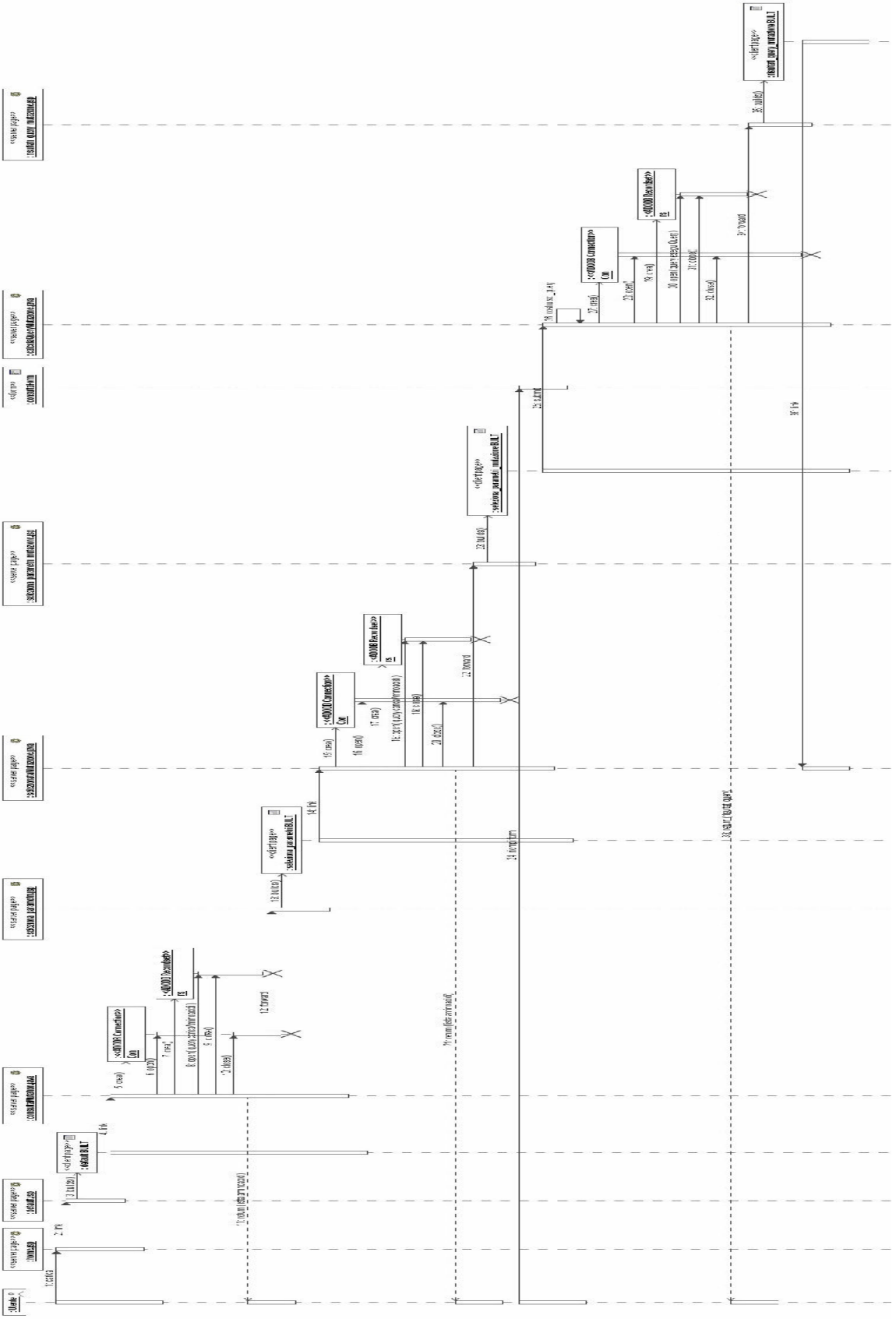
Caso d'uso: Visualizza link utili



Caso d'uso: Consultazione informazioni sull'intera proteina



Caso d'uso: Consultazione informazioni sulle mutazioni



APPENDICE C

LA PIATTAFORMA J2EE

Questa appendice illustra gli aspetti principali della piattaforma J2EE. Vengono illustrate le difficoltà che bisogna affrontare quando viene sviluppata una applicazione di tipo "Enterprise" e i vantaggi che la piattaforma fornisce per risolverle.

C.1 Introduzione alla Piattaforma

La Java 2 Platform Enterprise Edition è una piattaforma di sviluppo e deployment di applicazioni distribuite multi-livello con una forte orientazione enterprise, o aziendale.

Essa poggia le sue basi sulla piattaforma precedente realizzata dalla Sun, la J2SE (Java 2 Standard Edition) orientata allo sviluppo e deployment di applicazioni desktop tradizionali, aggiungendone numerose funzionalità(Fig.C.1).

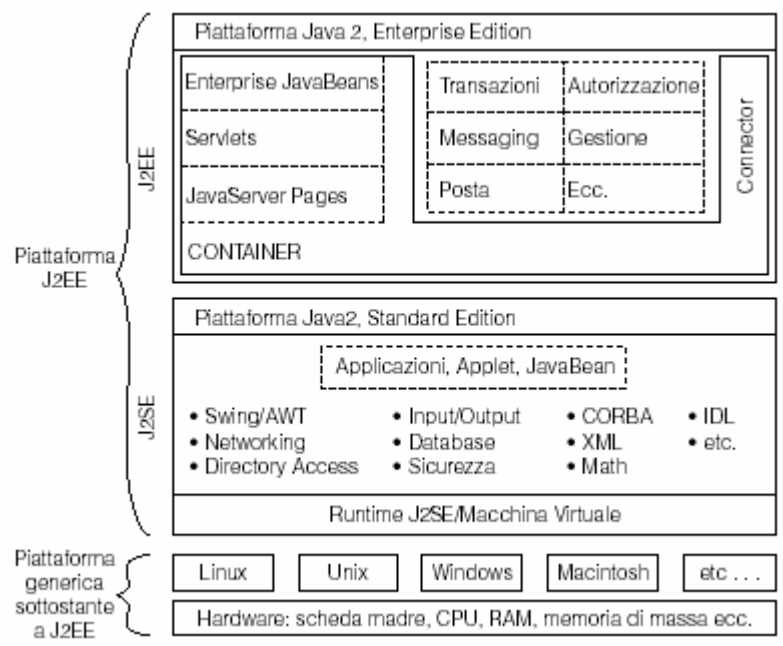


Figura C.1 - Funzionalità aggiunte dalla J2EE alla precedente J2SE

L'architettura proposta dalla piattaforma J2EE divide le applicazioni enterprise in tre strati applicativi fondamentali: componenti, contenitori e connettori.

Il modello di programmazione prevede lo sviluppo di soluzioni utilizzando componenti a supporto delle quali fornisce tre tecnologie fondamentali: EJB, Servlet e Jsp. I contenitori fungono da supporto ai componenti, in quanto forniscono l'ambiente

in cui i componenti vengono eseguiti. I connettori hanno il compito di preservare e proteggere investimenti in tecnologie già esistenti fornendo uno strato di connessione verso applicazioni-server o middleware di varia natura, come ad esempio i database relazionali a cui si accede tramite i driver JDBC.

Gli Application Server compatibili con tale tecnologia riuniscono tutti e tre questi strati in un'unica piattaforma standard, consentendo di sviluppare componenti in grado di girare in qualunque container compatibile con J2EE indipendentemente dal fornitore di software.

C.2 La Complessita' delle Applicazioni "Enterprise"

Durante il processo di sviluppo di applicazioni enterprise, gli sviluppatori devono affrontare molteplici difficoltà. Le principali sono rappresentate da:

- *Stabilità*, con tale termine si indica la capacità di una applicazione di comportarsi come previsto, senza subire crash;

- *Scalabilità*, è l'attitudine di un'applicazione a poter essere facilmente ampliata, soddisfacendo eventuali incrementi di carico nelle richieste;

- *Sicurezza*, è il grado di protezione o vulnerabilità di un'applicazione da utilizzazioni non autorizzate;

- *Semplicità*, per tale proprietà è necessario fare una distinzione: per gli utenti finali, la semplicità di un'applicazione dipende fondamentalmente dall'interfaccia

utente con la quale interagiscono; nella prospettiva dello sviluppatore software, invece, il grado di semplicità dipende dalla possibilità di sviluppare o migliorare facilmente ed efficientemente l'applicazione, mantenendone la stabilità, la scalabilità e la sicurezza globali;

- *Integrazione,* ovvero la capacità di combinare tecnologie recenti e vecchie, adattandosi il più possibile ai sistemi già presenti migliorandone le funzionalità;

- *Portabilità*, ovvero la possibilità di realizzare applicazioni funzionanti in ambienti differenti.

C.3 I Vantaggi Forniti dalla Piattaforma J2EE

Si procede ad analizzare i vantaggi derivanti dall'utilizzo della piattaforma:

- *Impiego di codice riutilizzabile*, J2EE supporta un modello di sviluppo di applicazioni basate su componenti nel quale le applicazioni enterprise sono assemblate con componenti software riutilizzabili scritte nel linguaggio di programmazione Java;

- *Facilitazioni nelle operazioni di manutenzione e di estensione*, le componenti sono unità di funzionalità di programma auto-contenute che operano indipendentemente le une dalle altre. Le architetture basate su componenti

generano applicazioni modulari che sono relativamente semplici da mantenere, aggiornare ed estendere;

- *Supporto al packaging ed al deployment delle applicazioni*, invece di forzare delle regole rigide per il packaging e il deployment delle componenti, J2EE è intenzionalmente flessibile e lascia agli sviluppatori la libertà di creare le soluzioni che meglio soddisfano le loro esigenze specifiche. Tali operazioni possono essere fatte su singole componenti o su gruppi;

- *Configurazione del comportamento nella fase di deployment*, il comportamento dell'applicazione può essere definito da programma tramite un opportuno codice contenuto nella componente o in modo dichiarativo tramite istruzioni statiche (dichiarazioni) nei file del *descrittore di deployment(web.xml);*

- *Suddivisione del lavoro*, le componenti sono delle unità di funzionalità software che possono essere assemblate in librerie o in applicazioni complete, cosa che favorisce una divisione precisa dei compiti;

- *Supporto per la scalabilità*, le componenti J2EE sono sempre eseguite nel contesto di un *container* che semplifica e automatizza lo *scaling* ("adattamento delle dimensioni") delle applicazioni distribuite senza richiedere alcuno sforzo da parte del programmatore;

- *Supporto per la sicurezza,* J2EE supporta la *sicurezza da programma,* che consente agli sviluppatori di codificare esplicitamente i controlli di autenticazione degli utenti e i controlli di accesso nelle componenti, e la

sicurezza dichiarativa role-based che può essere specificata all'atto del deployment di un'applicazione. Mentre i controlli di sicurezza da programma possono verificare gli accessi all'applicazione su base individuale, la sicurezza role-based permette a gruppi di individui di condividere i medesimi privilegi di accesso;

- *Stabilità e affidabilità*, dovute alle solide basi offerte dalla piattaforma sottostante J2SE e dal linguaggio di programmazione Java con il quale è realizzata;

- *Portabilità*, i programmi Java consistono in codice "bytecode" indipendente dalla specifica architettura hardware, eseguito nella *macchina virtuale Java* (JVM Java Virtual Machine). A runtime, le JVM convertono dinamicamente il bytecode nel codice macchina nativo richiesto dal particolare computer. Di conseguenza, qualsiasi computer che abbia una JVM adeguata può eseguire i programmi Java;

- *Integrazione con sistemi precedenti*, è disponibile un vasto assortimento di API standard che rendono possibile l'integrazione con sistemi esistenti.

C.4 Architettura del Sistema

La piattaforma J2EE è fondata su un'architettura multilivello nella quale le varie componenti che compongono un'applicazione vengono logicamente separate e distribuite su diversi livelli o strati dell'ambiente di elaborazione di rete.

Uno dei principali punti di forza della piattaforma è di aver abbandonato per i sistemi client-server l'architettura 2-tier per passare ad una multi-tier. Questo significa che se prima la maggior parte del carico pesava sul client e quindi le prestazioni andavano di pari passo con le risorse del PC, ora la maggior parte del lavoro viene spostata sul server, snellendo enormemente il carico del client e permettendo così di ampliare e rafforzare le funzionalità dell'applicazione

L'architettura 2-tier limita troppo lo sviluppo di applicazioni, soprattutto di quelle enterprise, nasce quindi un nuovo tipo di struttura in grado di supportare la suddivisione logica di un'applicazione: **Presentation, Business e Data Layer**. La prima area rappresenta l'interfaccia utente, la seconda parte realizza la logica e infine, il Data Layer contiene i dati necessari all'applicazione. Questo tipo di suddivisione dovrebbe facilitare la comprensione del salto che si compie passando ad un'architettura n-Tier. La logica dell'applicazione può essere maggiormente differenziata e suddivisa in base alla sua funzionalità.

Viene di seguito mostrato un esempio dei 3-tier, in cui una applicazione enterprise è tipicamente suddivisa(Fig. C.2)

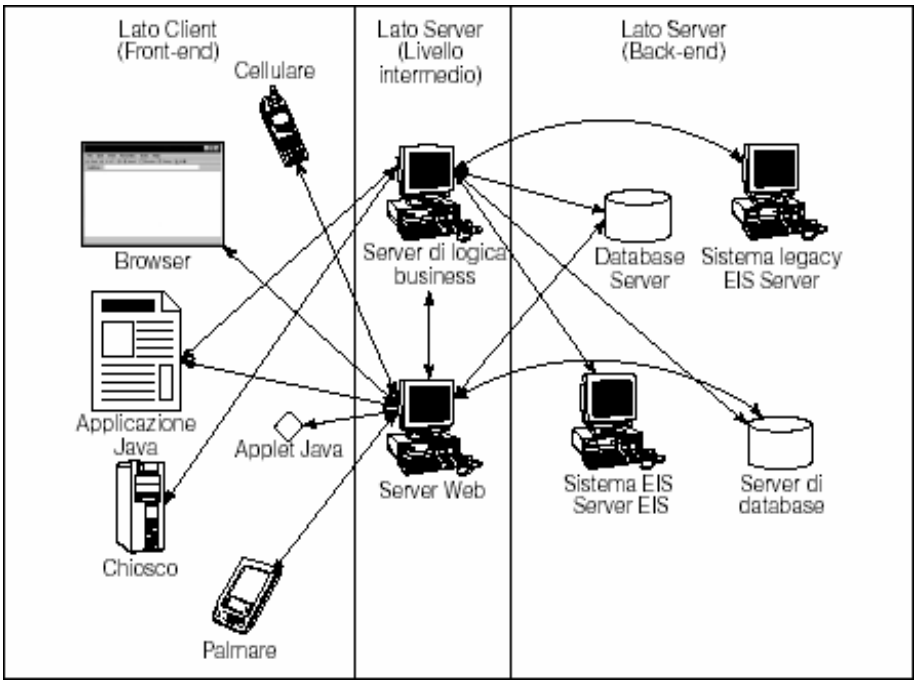


Figura C.2 - Livelli di una tipica applicazione enterprise

Ringraziamenti

Finalmente sono giunto al termine di questo percorso, che fatica, ma soprattutto che soddisfazione!!! Al termine di questo mio ultimo lavoro devo sentitamente ringraziare il prof. d'Acierno, sempre disponibile e pronto a dispensare consigli, nonché dell'avermi dato la possibilità di usufruire del suo studio(!!!), il dott. Facchiano Angelo e la dott.sa Marabotti Anna, per l'enorme aiuto negli aspetti bio-chimici della tesi, ed in generale tutti i componenti del Laboratorio di Bioinformatica del ISA-CNR di Avellino.

Un saluto e un grazie a tutti coloro che con me hanno condiviso l'intero percorso universitario, in particolar modo Michele, Gianrico, Crescenzo, Sergio e Rossella. Ricorderò sempre con immenso piacere i momenti passati con voi.

Ovviamente il ringraziamento più grande va a mio padre, a mia madre, a mio fratello Giuseppe e al mio amore Veronica, che mi hanno sempre supportato…e sopportato in tutti questi anni ed in particolare in quest'ultimo periodo.

Un ultimo pensiero va a mio nonno Michele, laureandomi credo di aver realizzato il suo desiderio più grande. Avrei tanto voluto averlo accanto in questo momento, come lo è stato in tanti altri momenti della mia vita.

Bibliografia

[1] ASMME, "*Associazione Studio Malattie Metaboliche Ereditarie"*, http://www.cometaasmme.org.

[2] SISMME, "*Società Italiana per lo Studio delle Malattie Metaboliche Ereditarie",* http://www.sismme.it.

[3] APMMC, "*Associazione Prevenzione Malattie Metaboliche Congenite"*, http://www.apmmc.it.

[4] Scriver C.R., Beandat A.L., Siley W.S., Valle D*., "The metabolic and molecular bases of inherited disease"*, McGraw-Hill, 2001.

[5] AIDMR, "*Agenzia Italiana Documentazione Malattie Rare"*, http://www.aidweb.org.

[6] NORD, "*National Organization for Rare Diseases"*, http://www.rarediseases.org.

[7] *"Adult Metabolic Transition Project",* http://depts.washington.edu.

[8] *"OMIM Database"*, http://www3.ncbi.nlm.nih.gov/entrez.

[9] Martin A.C.R., Facchiano A.M., Cuff A.L., Hernandez-Boussard T., Olivier M., Hainaut P., Thornton J. M., *"Integrating Mutation Data and Structural Analysis of the TP53 Tumor-Suppressor Protein"*, Wiley-Liss, 2002.

[10] Elsas II L.J., MD, and Kent Lai, PhD, "The molecular biology of galactosemia", Genetics in Medicine, 1998

[11] *"Codons and Aminoacids",* http://www.genomic.unimelb.edu.au/mdi/mutnomen/codon.html, gennaio 2005.

[12] *"Apache Software Foundation"*, http://www.apache.org

[13] *"Apache Jakarta Project"*, http://jakarta.apache.org.

[14] *"Apache T*omcat", http://jakarta.apache.org/tomcat.

[15] Brittain J., Darwin I.F., *"Tomcat: The definitive Guide"*, O'Reilly, 2003.

[16] *"PostgreSQL Database"*, http://www.postgresql.org, versione 7.3.

[17] Brookins A., Holloway M., *"Practical PostgreSQL"*, O'Reilly, 2001.

[18] Douglas K., Douglas S., *"PostgreSQL: The comprehensive guide to building, programming, and administering PostgeSQL databases"*, Second Edition, Luglio 2005.

[19] Booch G., Rumbaugh J., Jacobson I., *"The Unified Modeling Language User Guide"*, Addison-Wesley.

[20] Conallen J., *"Building Web Application with UML"*, Addison-Wesley, 1999.

[21] Fowler M., Scott K. *"UML Distilled"*, Addison-Wesley, 2000.

[22] Di Lucca G.A., Slide del corso di Ingegneria del Software, a.a. 2002/2003.

[23] Atzeni P., Ceri S., Paraboschi S, Torlone R., "Basi di dati" , Seconda Edizione, McGraw-Hill.

[24] *"Java 2 Platform Enterprise Edition",* http://java.sun.com/j2ee/.

[25] Marini F., *"J2EE: Il libro", 2002.*

[26] Johnson R., *"J2EE Design and Development"*, Wrox, 2002.

[27] Eckel B., "Thinking in Java 3rd Edition Revision 4.0", www.BruceEckel.com, novembre 2002.

[28] Hall M., “Core, Servlets and JavaServerPages”, Prentice Hall and Sun Microsystems.

[29] *“Progetto Jakarta Struts”*, http://struts.apache.org/.

[30] *“Exadel Studio Official Site”*, www.exadel.com.

[31] *“Eclipse Platform Official Site”*, www.eclipse.org.

[32] *“Exadel Studio 2.5 Getting Started Guide for Creating a Struts Application”*, ESStrutsGettingStarted.pdf, settembre 2005

[33] [www.mokabyte.it] A. La rotonda, Sviluppare applicazioni J2EE con Jakarta Struts, 2004.

[34] Dionisi S., *“Guida al framework Struts”*, http://www.javaportal.it/docs/struts1.htm.

[35] Cavaness C., *"Programming Jakarta Struts"*, 2nd Edition, O'Reilly, 2004.

[36] Goodwill J., Hightower R., *“Professional Jakarta Struts"*, Wrox, 2004.

[37] Siggelkow B., *“Jakarta Struts CookBook”*, O'Reilly, 2005.

[38] Robinson M., Finkelstein E., *“Jakarta Struts for Dummies”*, Wiley Publishing, 2004.

[39] Husted T., Dumoulin C., Franciscus G., Winterfeldt D., *"Struts in Action"*, Manning, 2003.

[40] Spielman S., *"THE STRUTS FRAMEWORK: Practical Guide for Java Programmers"*, Morgan Kaufmann Publishers, 2003.

[41] *"Il Framework Validator"*, http://jakarta.apache.org/commons/validator.

[42] Cavaness C., slide "The Validator Framework".

[43] JavaMail API, http://java.sun.com/products/javamail/index.html.

[44] Thomas S., *"SSL and TLS essentials: Securing the Web"*, Wiley Computer Publishing, 1999.

[45] Ditlinger S. (e altri), *"SSL Extension to Struts (SSLEXT)"*, http://sslext.sourceforge.net.

[46] "*GALT data.xls*", Laboratorio di Bioinformatica, ISA CNR Avellino