Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea in Scienze di Internet

APACHE STRUTS 2 PER LA REALIZZAZIONE DI MODULI ERP: COSTI E PROSPETTIVE

Tesi di Laurea in Programmazione Internet

Relatore:
Chiar.mo Prof.
ANTONIO MESSINA

Presentata da:
ROSSI CLAUDIO

III Sessione
Anno Accademico 2009/2010

Dedicato a tutte le persone che
hanno permesso il raggiungimento
di questo traguardo

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

L'evoluzione è una parte fondamentale di ogni sistema informativo, ma purtroppo è anche una delle fasi a maggior rischio e dall'investimento più oneroso. Quindi ogni azienda deve ponderare bene come e quando avviare questa fase. L'ambiente industriale italiano poi è un ambiente restio al cambiamento e poco incline alla spesa infatti tra i paesi occidentali si colloca all'ultimo posto in termini di risorse investite in ricerca e sviluppo [2]. Sulla base di queste premesse lo studio di fattibilità si colloca come uno step fondamentale per la programmazione di nuovi progetti, infatti lo scopo principale di quest'ultimo è la stima dei costi di realizzo di un progetto. Tutto ciò ne fa uno strumento importante per la vita aziendale che è sempre attenta agli sprechi e ai costi; quindi la redazione di un esauriente studio di fattibilità aiuterà l'eventuale sviluppo del progetto.
Questo documento tratta la redazione di uno studio di fattibilità per la realizzazione di un modulo integrativo di un ERP [27], per un'azienda umbra che produce abbigliamento d'alta moda. L'azienda è già in possesso di un ERP con il quale gestisce e monitora molte attivivià per esempio la fatturazione o le spedizioni; questo programma è stato sviluppato con l'ausilio del framework open source Apache Struts [9]. Siccome questa tecnologia è ormai datata il nostro team ha deciso di aggiornarsi e per sviluppare questo nuovo modulo, che prevede la gestione delle vendite interne ai dipendenti, si è deciso di adottare una tecnologia più innovativa. Questo nuovo modulo dovrà essere fruibile come web application e dovrà essere innestato nel vecchio progetto come nuovo punto di menù.

Come nuova tecnologia implementativo si è scelto il framework Apache Struts 2 [10], naturale evoluzione di Struts, pensando, in tal modo, di limitare i costi e quindi i tempi della nostra formazione.
Questa tesi vuole trattare sia l'aspetto economico che l'aspetto implementativo del progetto. Si cercherà di mostrare i costi del progetto attraverso un preventivo dei costi di realizzo, creato con l'ausilio del metodo COCOMO [25], e dei tempi di studio e implementazione con l'aiuto di appositi report, come il diagramma di Gantt [26].
Nel corpo centrale della relazione si mostreranno le idee di progettazione delle funzioni, i problemi da affrontare e i casi d'uso riscontrati.
Negli ultimi capitoli inoltre si esporrà un piccolo dimostratore delle funzioni del applicativo con l'apporto di screenshot e parti di codice finalizzate a mostrare la completezza del lavoro svolto e le conoscenze acquisite nel suo realizzo.
Va sottolineato il fatto che il progetto nasce come progetto pilota per stabilire la convenienza da parte di un gruppo di lavoro di passare alla nuova tecnologia o di cercare un'altra strada.

Capitolo 1

Tecnologie per lo sviluppo di web application

1.1 Evoluzione da applet ai framework open source

La storia delle web application nasce nel 1993 con il primo linguaggio di scripting. È un linguaggio server-side che quando il client richiede un programma CGI il server esegue a tempo reale il programma e restituisce le informazioni tramite protocollo HTTP. [19]
Nel 1995 nasce un nuovo modo di programmare siti dinamici nel web e sono le Java applet. Queste sono programmi scritti in linguagggio Java che possono essere letti dai web browser utilizzanti la JVM. Sono scritte per creare siti dinamici e rendere interattivo o creare piccole funzioni all'interno di un sito. [16]
Nel 1997 dal lavoro di Brendan Eich della Netscape Communications nasce JavaScript [22]. JavaScript è uno standard ISO. La caratteristica principale è quella che in JavaScript il codice non viene compilato, ma interpretato (in JavaScript lato client, l'interprete è incluso nel browser che si sta utilizzando). La sintassi è relativamente simile a quella del C, del C++ e del Java. Il linguaggio definisce le funzionalità tipiche dei linguaggi di programmazione

ad alto livello (strutture di controllo, cicli, ecc.) e consente l'utilizzo del paradigma object oriented. Altri aspetti di interesse: in JavaScript lato client, il codice viene eseguito direttamente sul client e non sul server. Il vantaggio di questo approccio è che, anche con la presenza di script particolarmente complessi, il server non viene sovraccaricato a causa delle richieste dei clients. Di contro, nel caso di script che presentino un sorgente particolarmente grande, il tempo per lo scaricamento può diventare abbastanza lungo. Un altro svantaggio è il seguente: ogni informazione che presuppone un accesso a dati memorizzati in un database remoto deve essere rimandata ad un linguaggio che effettui esplicitamente la transazione, per poi restituire i risultati ad una o più variabili JavaScript; operazioni del genere richiedono il caricamento della pagina stessa. Con l'avvento di AJAX tutti questi limiti sono stati superati. [22]

Le tecnologia fino ad ora descritte però non permettevano di creare applicazioni complesse, per far questo dobbiamo aspettare il 2003 e l'avvento delle Java Servlet. Le Java Servlet sono uno standard creato nel 1996 da Sun Microsystem di cui dal 2010 siamo alla versione 3.0. Ma cosa è una Servlet? Una Servlet è una classe Java conforme alle Java Servlet API che permette di rispondere a richieste HTTP.

Con questo mezzo gli sviluppatori possono utilizzzare codice Java per aggiungere contenuto dinamico ad un server web. In particolare una Servlet è un oggetto che riceve una richiesta HTTP e genera una risposta sulla base di tale richiesta. I vantaggi di utilizzare una Servlet rispetto a funzioni CGI sono migliori prestazioni in velocità e facilità d'uso. Infatti con CGI ogni richiesta creava un nuovo processo e questo poteva far sorgere un problema di overhead di processi in special modo se le richieste erano tutte molto semplici, mentre con le Java Servlet questo problema viene risolto gestendo ogni richiesta con un thread separato all'interno del processo del web server. [21]

Con l'avvento delle Servlet e la possibilità di eseguire codice per la creazione di pagine dinamiche a lato server, sono nati i web application framework. Questi sono nati con lo scopo di alleggerire il lavoro associato allo sviluppo

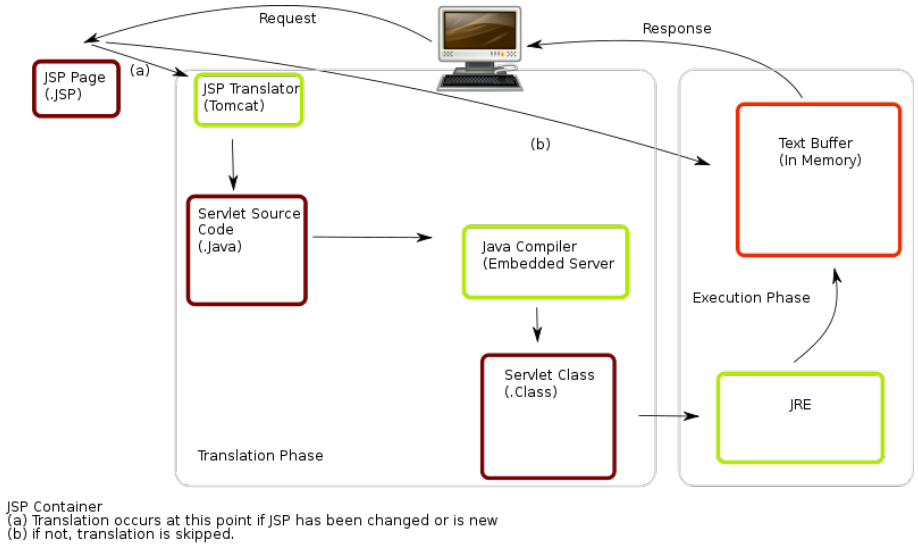

Figura 1.1: Vita di una Servlet. Da [21]

delle attività più comuni di una applicazione web. Molti framework, infatti, forniscono funzioni e codice per l'esecuzione di attività comuni a quasi tutte le applicazioni web come per esempio:

- accesso alle basi di dati
- creazione di template HTML
- gestione della sessione dell'utente e tanti altri.

Per riassumere lo spirito di un web application framework è nato l'acronimo DRY (Don't Repeat Yourself), nel senso che viene fortemente promossa la riutilizzazione del codice. [24] Nel 2005 si ha una nuova evoluzione, nasce AJAX acronimo di Asynchronous JavaScript and XML. Questa tecnica di sviluppo ha portato numerose migliorie e possibilità di sviluppo come il poter ricaricare una parte di una pagina senza una esplicita richiesta dell'utente. AJAX, infatti, esegue le sue richieste al server in modo asincrono nel senso, che i dati extra sono richiesti al server e caricati in background senza interferire con il comportamento della pagina esistente. AJAX per il suo funzionamento utilizza una combinazione di:

- HTML e CSS per il markup e lo stile
- DOM manipolato con JavaScript o altri linguaggi di scripting per mostrare il risultato dei dati
- l'oggetto XMLHttpRequest per la comunicazione e lo scambio di dati da e per il web server

Con l'avvento di AJAX le applicazioni si sono fatte più veloci perchè, quando utilizzata l'applicazione può inviare al server richieste per solo una parte della pagina, allegerendo in tal modo la quantità di dati da trasmetere. [17]

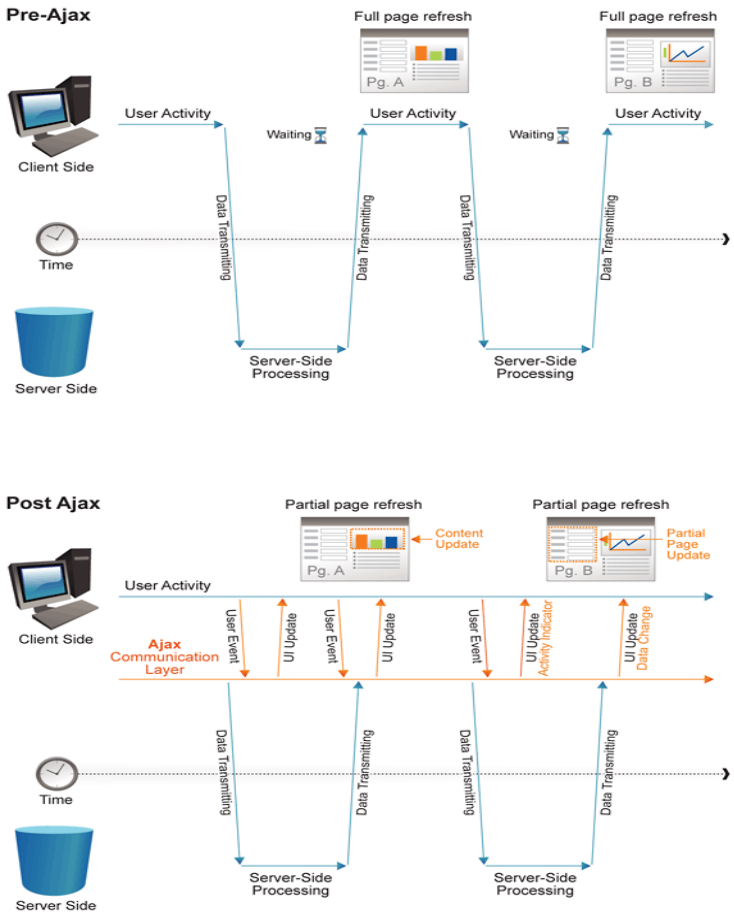

Figura 1.2: Applicazione con AJAX. Da [15]

1.2 Struts e MVC

1.2.1 MVC Model-View-Controller

Il pattern MVC è un pattern architetturale, molto diffuso utilizzato nell'ingegneria del software. Originariamente utilizzato dal linguaggio Smalltalk al giorno d'oggi è utilizzato da tutti i maggiori linguaggi di programmazione. Il pattern è basato sulla separazione dei compiti tra i tre componenti che ne fanno parte:

- il Model che racchiude il core dell'applicazione, e quindi i metodi per accedere ai dati ed esaudire le richieste dell'utente
- il View che si occupa della visualizzazione dei dati elaborati dal Model
- il Controller che riceve le richieste dal View e cerca nel Model l'azione da eseguire. [23]

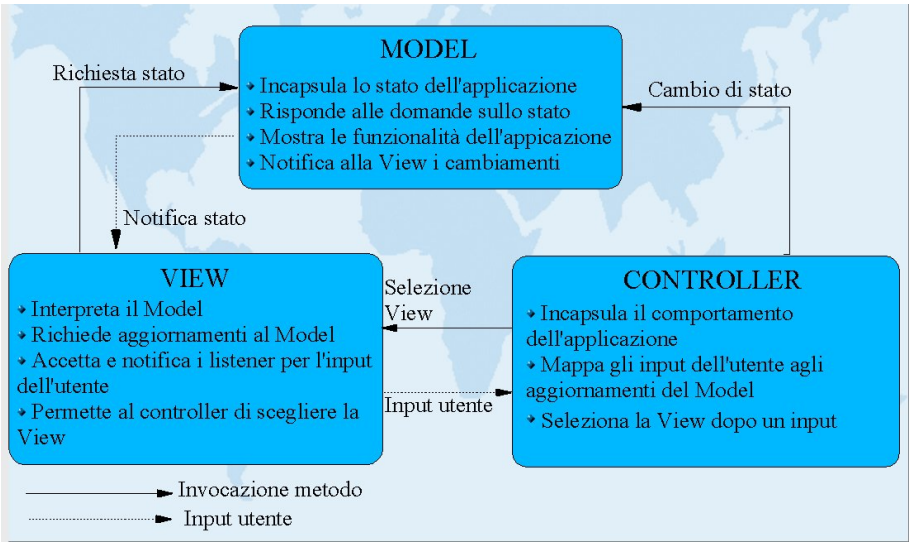

Figura 1.3: MVC pattern. Da [5]

1.2.2 Struts

Struts è un framework open source creato dalla Apache Software Foundation come implementazione Java server side del pattern MVC. Il progetto nasce in seno ad IBM che nel 2000 lo dona alla Apache Software Fondation. [18] Dal 2003, dopo il rilascio della versione 1.1, Struts diventa uno dei framework di sviluppo per applicazione web di maggior utilizzo. [3] Alcune delle caratteristiche che hanno fatto diventare Struts un framework leader sono:

- 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
- Rapidità di sviluppo: è possibile sviluppare in parallelo le varie parti dell'applicazione, logica di business e di view. [3]

Il componente più importante del modello è il controller al quale è affidato il compito di instradare le richieste dell'utente e di restituirgli i dati richiesti. Il Model, il cui compito è la persistenza dei dati, può utilizzare diverse tecnologie con cui implementare questa funzione, come ORM o Hibernate, anche il View non è vincolato ad un sola tecnologia implementativa ma si può utilizzare la tecnica che si preferisce come per esempio JSP o Velocity. [3]
Struts fornisce gli strumenti per la creazione di applicazioni web;
i componenti più importanti del framework sono:

- la *ActionServlet* che ricrea il funzionamento di una Servlet e gestisce tutte le richieste. Estende la classe Javax.Servlet.http.HttppServlet
- lo Struts-config.xml è il cuore dell'applicazione, al suo interno vengono definiti tutti gli elementi dell'applicazione. Inoltre vengono mappate le azioni dell'applicazione e viene utilizzato dal controller per smistare le richieste.

- le *Action* sono le classi alle quali la ActionServlet delega l'esecuzione delle richieste dell'applicazione
- le *ActionForm* sono dei veri contenitori di dati. Fanno riferimento ad uno specifico form e vengono popolati automaticamente dal framework con i dati contenuti nella request HTTP
- le *ActionMapping* contengono gli oggetti associati ad una Action nello Struts-config come ad esempio gli ActionForward
- le *ActionForward* contengono i path ai quali la Servlet di Struts inoltra il flusso in base alla logica dell'applicazione
- i $Custom - tags$ sono tag particolari forniti dal framework Struts per assolvere a molti dei più comuni compiti delle pagine JSP. [4]

Ciclo di vita

Gli oggetti fondamentali del framework Struts sono le azioni che vengono invocate dal controller dopo un evento avvenuto nel view. Ogni azione segue un preciso percorso da quando viene richiesta tramite l'$HTTPRequest$ a quando viene restituita attraverso lo $HTTPResponse$.
In fase di start up il controller legge il file Struts-config.xml in cui sono mappati tutti i path delle azioni. Fatto questo il controller si mette in attesa di una richiesta proveniente dall'esterno. Ogni richiesta del cliente arriva generalmente tramite protocollo HTTP, incapsulata in un oggetto $HTTPRequest$. La richiesta viene intercettata dalla Servlet che in automatico popola l'ActionForm e l'ActionMapping relativi all'azione richiesta. Inoltre la Servlet richiama la Action delegata all'esecuzione dell'azione relativa al path richiesto. Questa eseguirà l'azione e, se sono richiesti dei dati dal DB, interpellerà il Model. Al termine dell'esecuzione la Action restituisce alla Servlet una ActionForward contenente il path della vista da restituire all'utente. Infine la Servlet esegue il forward della ActionForward specificata incapsulando la pagina in un oggetto $HTTPResponse$. [4]

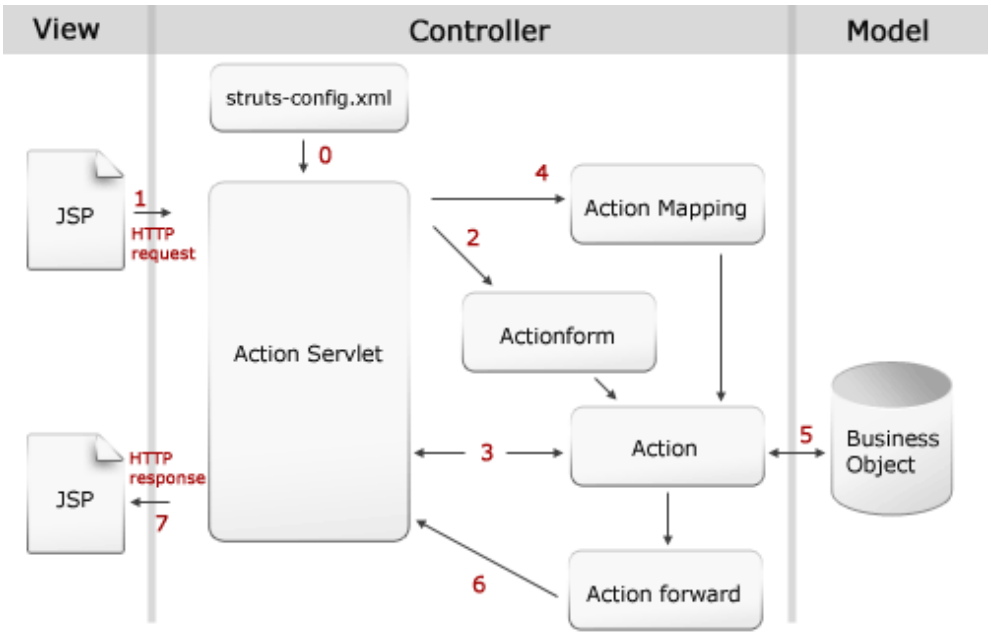

Figura 1.4: Ciclo di vita di Struts. Da [4]

1.2.3 Struts 2

Nel 2008 sempre dai laboratori della Apache Software Fondation nasce Struts 2. Struts 2 nasce dal'unione dei progetti Struts Ti e WebWork 2, il nome Struts è rimasto siccome tra i due era il più famoso. L'obiettivo di Struts 2 era di rendere più agevole lo sviluppo del programma. Per far ciò ha diminuito i tempi di configurazione dei file XML o attraverso impostazioni predefinite, o con l'utilizzo di annotazioni, che possono ridurre anche a zero le configurazioni XML. Ora le azioni sono POJO, acronimo di Plain Old Java Object, che in questo modo agevolano la loro testabilità e diminuisco l'accoppiamento dell'applicazione. Sempre per diminuire l'accoppiamento sono stati inseriti gli Interceptors che consentono un pre e un post controllo dei dati. Struts 2 è un progetto modulare perchè consente l'integrazione, attraverso plug-in, con altri framework come: Hibernate, Spring Framework, Tiles 2, Google Web Toolkit (GWT), Dojo Toolkit e altri. [14]

Come il predecessore Struts 2 consente l'utilizzo della tecnologia che si

preferisce per la parte di View. Gli sviluppatori hanno anche dotato Struts 2 di una custom tag propria che consente di ricreare nelle pagine di View molte funzioni integrando perfettamente AJAX. Come Struts il 2 è basato sul concetto di azione quindi non stravolge l'idea originale e consente in breve tempo ad un programmatore proveniente dal primo framework di entrare nella sua logica. Una differenza importante comunque è che Struts 2 è un framework pull-MVC, ciò vuol dire che la parte view ha la capacità di estrarre i dati direttamente dalla Action senza la necessità di un oggetto separato. [13]

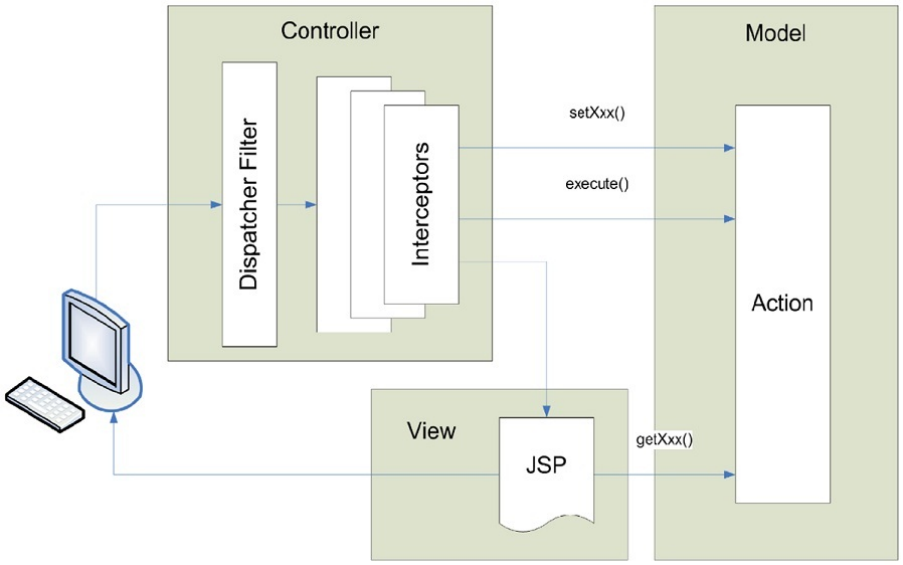

Figura 1.5: Struttura di Struts 2. Da [13]

Ciclo di vita di Struts 2

In Struts 2 ogni azione è associata ad un url che spesso è similare a `http://localhost:8080/app/index.action`, il pattern *.action serve ad identificare una richiesta che il FilterDispatcher prenderà a carico. Mentre in Struts la *ActionServlet* era il cuore dell'applicazione, in Struts 2 questa posizione viene ereditata dal *FilterDispatcher* che ha il compito di intercettare tutte le richieste e indirizzarle verso l'azione ad esse associate. Il

FilterDispatcher oltre ad avere il compito di intercettare e dirigere le azioni ricopre anche il ruolo di configuratore in fase di startup dell'applicazione, cioè è suo compito:

- Inizializzare i contenuti statici del server, come i DOJO e gli script in JavaScript e tutti i file configurati dall'utente
- Inizializzare il *ConfigurationManager* e gli *ActionMapper*, che serviranno in seguito per instradare le richieste verso le giuste azioni
- Creare i contesti di azione
- Creare gli *ActionProxy*, queste classi contengono le configurazioni e le informazioni di contesto per eseguire le richieste e dopo l'esecuzione conterranno al loro interno il risultato che è stato processato
- Eseguire il cleanup, per assicurare il buon funzionamento dell'applicazione esegue automaticamente il cleanup degli oggetti *ActionContext*

Come mostra il diagramma di sequenza ogni azione richiesta dall'utente viene intercettata dal*FilterDispatcher* che chiama la classe *ActionInvocation*. Questa classe gestisce l'ambiente di esecuzione e contiene lo stato della conversazione dell'azione in fase di elaborazione. Questa classe è il nucleo della classe *ActionProxy*. L'ambiente di esecuzione è composto da tre componenti: le azioni, gli interceptors e i risultati. Uno dei compiti della *ActionInvocation* è invocare i metodi delle azioni al momento opportuno. Un altro è creare le istanze delle azioni, al contrario della versione precedente in Struts 2 ogni richiesta prevede la creazione di un nuovo oggetto azione. Questo può creare un lieve calo prestazionale dell'applicazione, ma concede il vantaggio di trattare questi oggetti come dei POJO.
Dopo l'esecuzione della *ActionInvocation* si entra nel core dell'applicazione cioè si invoca l'azione e si esegue la logica di business prevista per la richiesta effettuata. Come detto, in Struts 2 il core dell'applicazione sono le azioni, una classe Java per essere definita azione deve soddisfare il seguente requisito:

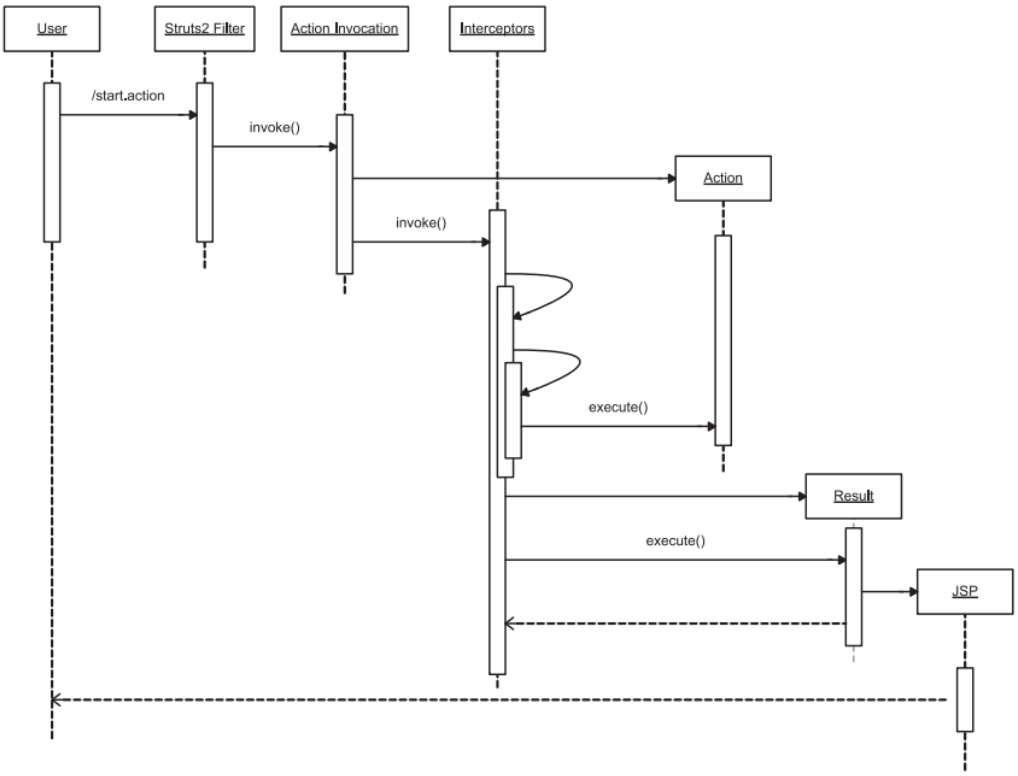

Figura 1.6: Ciclo di vita di Struts. Da [14]

- possedere un metodo a firma vuota che restituisca una stringa o un oggetto *Result*.

Di default questo metodo viene chiamato *execute*(). Quando il risultato è una stringa il corrispondente oggetto *Result* viene ricercato nella mappatura e istanziato. In Struts 2 non è più necessario che un'azione estenda un azione preconfigurata fornita dal framework. Nonostante questo, Struts 2 fornisce allo sviluppatore alcune interfacce con cui implementare le azioni sviluppate, questo sono *Action* e *ActionSupport*. [14] All'interno del framework oltre il filtro e le azioni sono presenti anche altri oggetti, gli *interceptors* che concedono la possibilità di rendere più snello e leggibile il codice di core delle azioni lasciando agli *interceptors* le funzioni comuni a tutte le azioni come per esempio la validazione dell'input. Usando gli *interceptors* si può:

- fornire logica pre-configurazione, prima della chiamata dell'azione
- fornire logica post-configurazione, dopo l'esecuzione dell'azione
- catturare le eccezioni in modo che possa essere eseguito codice alternativo o possa essere restituito un risultato diverso.

Struts 2 fornisce al suo interno una serie di *interceptors* già configurati e pronti per essere utilizzati, oltre a questi è possibile scriverne di nuovi: basta creare un interfaccia che estenda l'interfaccia *Interceptor*. [14]

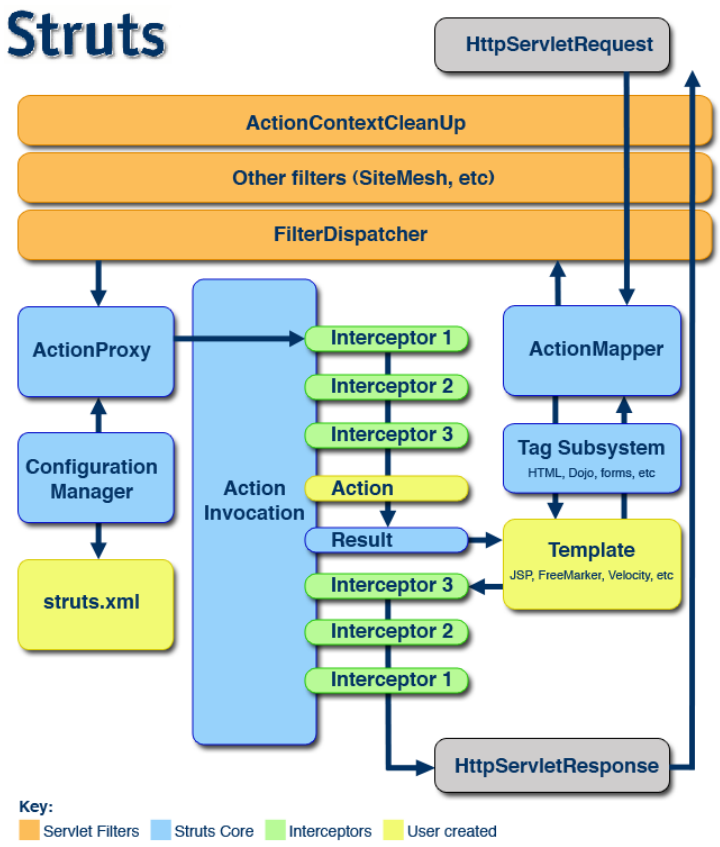

Figura 1.7: Architettura di Struts 2. Da [11]

Differenze tra Struts e Struts 2

La nuova versione di Struts presenta alcune differenze importanti dalla precedente che sono riassunte nella tabella 2.4.

	Struts	**Struts 2**
Azioni	Struts fornisce classi astratte per l'estenzione dei metodi comuni nelle azioni implementate dal programmatore	Struts fornisce interfaccie astratte per l'implementazione dei metodi comuni nelle azioni implementate dal programmatore
Threading	Le classi sono istanze singleton. Il server quindi adotta una strategia thread-safe per la sincronizzazione delle richieste	Le azioni sono istanziate ad ogni richiesta quindi non ci sono problemi di thread-safe.
Servlet	Le azioni sono dipendenti dalla Servlet in quanto gli oggetti *HTTPResponse* e *HTTPRequest* sono passati al metodo execute quando viene chiamata un'azione	L'ambiente Servlet viene rappresentato attraverso una mappa questo permette il poter testare una azione singolarmente. Permette inoltre ad ogni azione di accedere agli oggetti *HTTPResponse* e *HTTPRequest*
Expression Language	Integra JSTL come EL.	Permette l'utilizzo di JSTL, ma integra anche il più potente OGNL
Binding dei valori	Utilizza il meccanismo standard per associare oggetti alla pagina	Utilizza un ValueStack che consente l'accesso ai valori senza accoppiamento ad un particolare oggetto. Quindi consente il riuso con oggetti che hanno proprietà omonime ma di tipo diverso

	Struts	**Struts 2**
TagLibrary	Sono disponibili diverse tag library, suddivise per tipo di argomento trattato (logic, bean, HTML)	È disponibile un'unica tag library che mette a disposizione sia operazioni di logica che di rendering HTML
Tipi di conversione	All'interno dei form le proprietà sono solitamente delle stringhe. Struts1 utilizza Commons-Beanutils per la conversione di tipo. I convertitori sono per classe, e non configurabili.	Struts2 utilizza OGNL per le conversioni di tipo. Include convertitori per i tipi di oggetti di base e i tipi primitivi, approfittando del auto-boxing di Java5.
web.xml	Il controllo viene affidato ad una Servlet. Di default alla ActionServlet ma è possibile definire una Servlet personalizzata	Il controllo viene affidato ad un Filter. Di default al FilterDispatcher ma è possibile definire un Filtro personalizzato
URI pattern	Di default viene utilizzato il pattern *.do per identificare una richiesta che la *ActionServlet* prenderà in carico	Di default viene utilizzato il pattern *.action per identificare una richiesta che il *FilterDispatcher* prenderà in carico
File di configurazione	Di default il nome del file di configurazione è Struts-config.xml che va posizionato allo stesso livello del file web.xml	Di default il nome del file di configurazione è struts.xml che va posizionato in una directory del classpath

	Struts	**Struts 2**
Input	Vengono utilizzati gli ActionForm per l'acquisizione degli input. Come ogni azione, tutti gli *ActionForm* devono estendere una classe base. Poichè i JavaBeans non possono essere usati come *ActionForm* si viene a creare una ridondanza di classi.	Vengono utilizzate direttamente le proprietà dell'azione come proprietà d'ingresso eliminando la necessità di un oggetto di input.
Mapping delle Action	Il mapping di una Action viene definito nel file di configurazione mediante il tag action-mapping	Il mapping di una Action viene automaticamente generato dalla concatenazione package-nome della action, entrambi definiti nel file di configurazione

Tabella 1.4: Differenza tra Struts e Struts 2 [1] [4]

Capitolo 2

Il progetto

Il contesto in cui si colloca questa tesi è un ERP (Enterprise Resource Planning) [27]. Un ERP è un sistema informativo che integra tutti i processi di business rilevanti di un'azienda (vendite, acquisti, gestione magazzino, contabilità ecc.) Con l'aumento della popolarità dell'ERP e la riduzione dei costi per l'ICT (Information and Communication Technology), si sono sviluppate applicazioni che aiutano i business manager ad implementare questa metodologia nelle attività di business come: controllo di inventari, tracciamento degli ordini, servizi per i clienti, finanza e risorse umane. La prima versione dell'ERP metteva in collegamento diretto l'area di gestione contabile con l'area di gestione logistica (magazzini ed approvvigionamento); successivamente si sono iniziate ad implementare le relazioni interne anche con le aree di vendita, distribuzione, produzione, manutenzione impianti, gestione dei progetti ecc. Un modulo di grande importanza in un ERP è il sistema di Pianificazione Fabbisogno Materiali o Materials Requirements Planning (MRP) e la sua evoluzione MRP II (integrati nel sistema ERP) che permettono di programmare logiche di ordini automatici ai fornitori veramente sofisticate, tanto da tener conto dei tempi di consegna e di messa in produzione del prodotto; questa metodologia permette di ottimizzare la rotazione dei materiali nei magazzini e la minimizzazione delle giacenze che impattano a livello contabile e fiscale. Da evidenziare anche la crescita, sullo

scenario nazionale, di ERP tutti italiani che garantiscono la gestione completa degli adempimenti contabili e fiscali rispetto alla complessa normativa italiana; questi ERP a differenza dei leader dello scenario internazionale si calano in maniera più precisa nel "*modus operandi*" dell'azienda italiana con conseguente minor sforzo di adattamento alle procedure delle aziende che li adottano. A tutt'oggi i moderni sistemi di ERP coprono tutte le aree che possono essere automatizzate e/o monitorate all'interno di un'azienda, permettendo così agli utilizzatori di operare in un contesto uniforme ed integrato, indipendentemente dall'area applicativa.
Il mercato degli ERP si scinde in due grandi macro aree, le grandi imprese e le multinazionali, e le PMI. Il primo ramo è dominato dai grandi produttori di ERP come SAP o ORACLE. Il secondo ramo vede una penetrazione minore dei grandi produttori a causa degli ingenti costi del software e della difficile costumizzazione e flessibilità del programma rilasciato, questo perciò favorisce i produttori locali e i piccoli team di sviluppo che consentono all'azienda di limitare i costi ed avere un prodotto creato su misura. Lo ERP, nel cui contesto si colloca il modulo sviluppato in questa tesi, si attesta come un ibrido tra le due situazioni siccome, l'azienda per la quale si è svolto questo lavoro è un'impresa in rapida crescita che sta diventando oramai una grande impresa. Nonostante la crescita, la filosofia aziendale in campo tecnologico è che il software si deve adattare al ciclo produttivo aziendale e non viceversa. Seguendo questa linea l'azienda ha continuato a utilizzare software sviluppato espressamente per le sue necessità e non si è mai affidata a software preconfezionato.
A questo scopo è nato un rapporto di lavoro duraturo di lavoro tra l'azienda e il team di sviluppo. Questo modello di business è conosciuto come esternalizzazione o outsourcing [28], questo modello nasce quando un'azienda decida di far ricorso al mercato per l'approvvigionamento di beni intermedi e/o servizi alla produzione quindi questo implica una qualche forma di stabilità di rapporto di collaborazione tra l'impresa e un terzista.
Questo modello presenta per l'impresa committente i seguenti vantaggi:

- aumento del livello di specializzazione nello svolgimento di certe attività
- rifocalizzazione sulle core competence o competenze distintive dell'impresa
- aumento della flessibilità dell'impresa, sia operativa che strategica

Comunque i contratti di outsourcing presentano anche alcuni svantaggi come:

- trattare con un'azienda esterna, che può, se ha un forte potere contrattuale, imporre prezzi o tempi di realizzo diversi da quelli aspettati
- non poter monitorare totalmente il processo produttivo

Per le aziende di servizi all'impresa, invece, i contratti di outsourcing sono un'importante fonte di lavoro, ma presentano anche alcuni svantaggi da cui devono sapersi tutelare. Lo svantaggio maggiore è il fidealizzarsi con un solo committente quindi in pratica dipendere dalle scelte di quest'ultimo. Per ovviare a questo handicap l'azienda partner dovrebbe lavorare per più di un singolo outsourcee in modo da poter sopravvivere a possibili rescissioni di contratto di un committente o possibili fallimenti aziendali di un suo partner commerciale.

2.1 Il modulo da sviluppare

L'azienda di recente ha richiesto l'implementazione di un nuovo modulo. Precisamente ha richiesto la possibilità di poter gestire e monitorare le vendite interne ai dipendenti. Questa richiesta è giunta in seguito alla necessità di dover tracciare questa attività, che prima veniva gestita solo in modo cartaceo e non lasciava traccia nei sistemi aziendali. Il progetto si suddivide in due macroaree:

- il database e la gestione dei dati
- la parte di statistica e estrazione dei dati.

Il software aziendale è un software eterogeneo: per la parte riguardante il database, la gestione dei dati e le funzioni di produzione viene utilizzato un sistema sviluppato con RPG, mentre i moduli del settore commerciale, amministrativo e di gestione statistica sono sviluppate in Java, in questa tesi si presenterà solo l'estrapolazione dei dati.
Per il progetto l'azienda ha richiesto queste funzioni principali:

- le ricerca di tutti i documenti aperti
- il dettaglio di una persona
- il totale venduto

La richiesta di questo modulo è pervenuta, perchè l'azienda sta vivendo un periodo di certificazione che gli permetterà di fregiarsi del titolo di made in italy certificato, quindi i certificatori hanno richiesto all'azienda di monitorare tutti i movimenti della merce, comprese le vendite interne ai dipendenti che non erano monitorate.

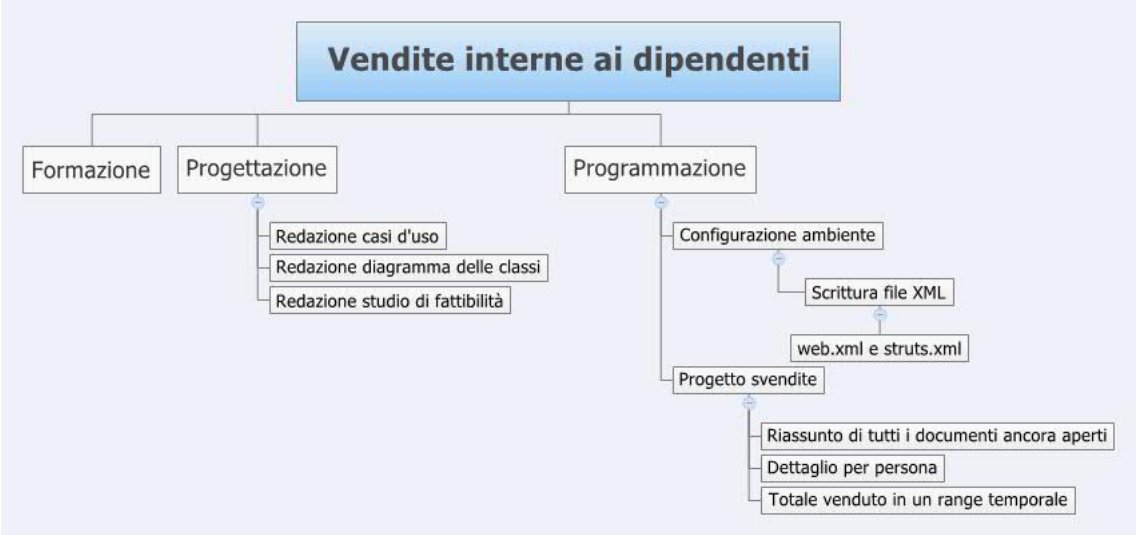

Figura 2.1: WBS

2.2 Analisi

La prima cosa da svolgere per la realizzazione di un buon progetto è la stesura di tutte le attività ad esso inerenti. Questo step aiuta notevolmente il lavoro del project manager perchè gli consente di aggregare in un unico grafico tutte le funzioni del progetto e avere in modo tale il quadro completo della situazione. Per far ciò vengono usate le Work Breakdown Structure (WBS, Struttura Analitica di Progetto) [29] che sono grafici che riuniscono tutte le attività di un progetto. In figura 2.1 è riportato la WBS del progetto "Vendite interne ai dipendenti".
Il progetto è composto da 3 grandi macro aeree che sono:

- formazione
- progettazione
- programmazione

Queste tre azioni formano il WBS di grado 1.
Le azioni di progettazione e programmazione comprendono a loro volta più attività e queste formano i WBS di grado 2 per le due azioni. Le attività dell'azione "progettazione"sono:

- redazione dei casi d'uso
- redazione del diagramma delle classi
- redazione dello studio di fattibilità

Le attività dell'azione "programmazione"sono:

- configurazione dell'ambiente, che prevede la stesura dei file xml
- progetto svendite[1], dove è prevista la realizzazione del codice per le funzioni commissionate

[1]Svendite interne ai dipendenti

2.2.1 Specifiche funzionali

Per ogni funzione è richiesta l'emissione di un foglio excel con i risultati restituiti dal database e la creazione di un punto di menù da cui richiamare la funzione.

Requisiti funzionali

I requisiti funzionali evidenziati sono:

- richiamare la funzione da un punto di menù
- presentare un form di inserimento
- accettare in input le richieste dell'utente
- restituire un file xls come risposta
- consentire di salvare nel proprio pc il file xls di risposta
- consentire di aprire il file xls di risposta

I requisiti non funzionali evidenziati sono:

- il file excel per i documenti aperti deve mostrare il nominativo, l'importo e il numero di capi in pendenza ad ogni persona
- il file excel di dettaglio deve mostrare tutti i capi in pendenza alla persona ricercata e il loro stato (aperto, venduto, reso)
- il file excel del totale venduto deve mostrare nel foglio 1 una riga per ogni tipologia, sia essa una data o un tipo filato, e per questa la sommatoria dei capi venduti, il totale incassato e la sommatoria del costo industriale. I fogli successivi devono riportare il dettaglio di ogni tipologia scritta nel foglio 1. Per facilitare la ricerca nei fogli successivi il nome di ogni foglio dovrà essere uguale alla descrizione scritta nel foglio dei totali
- l'accesso all'applicazione deve essere consentito solo ad alcuni utenti autorizzati

Casi d'uso

Nel progetto si individua un caso d'uso principale che riassume tutte le funzioni e che viene riportato in figura 2.2.

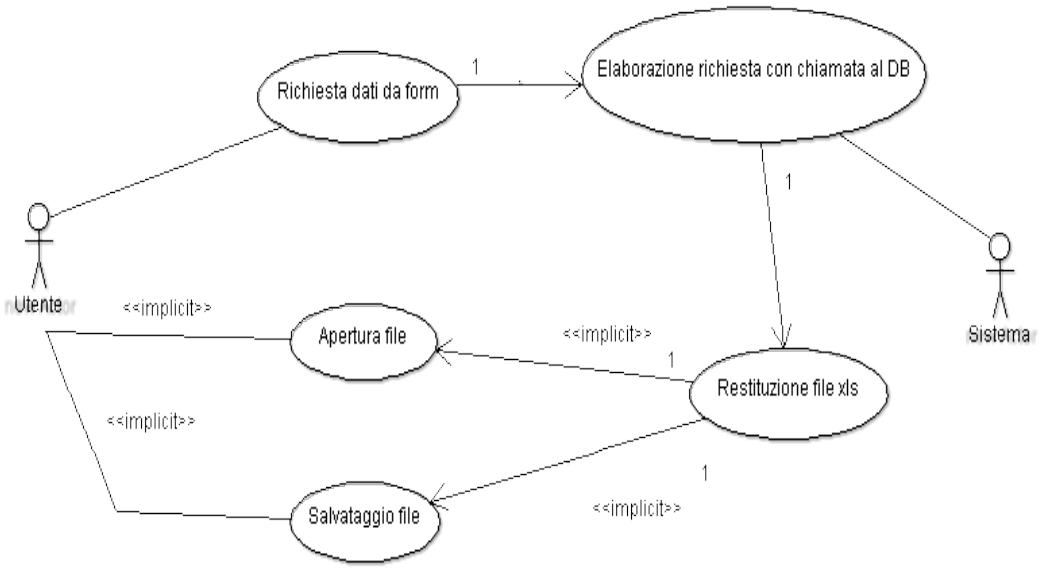

Figura 2.2: Diagramma dei casi d'uso

Esiste un solo caso d'uso poichè ogni funzione percorre uno schema logico unico cioè composto da tre fasi:

- l'utente clicca sul punto di menù il sistema risponde con il form di richiesta associato e si mette in attesa dell'utente;
- l'utente seleziona i criteri di ricerca e li spedisce al sistema che, elabora la richiesta e restituisce un foglio excel
- l'utente può scegliere se aprire o salvare nel proprio pc il file restituito

Tempistiche

Nell'ottica di un progetto un fattore importante è il tempo. Un buon project manager deve saper stimare il tempo necessario alla realizzazione del progetto dosando bene criteri di prudenza e difficoltà progettuali. Il diagramma maggiormente utilizzato per riportare questi dati è il diagramma di Gantt che consente di mostrare in una griglia tutte le operazione da svolgere e le stime dei loro tempi di realizzo.
In figura 2.3 è riportato il diagramma di questo progetto. Dopo una prima fase di studio e redazione di tutti i documenti e report necessari a supportare il codice e lo studio di fattibilità si può procedere alla implementazione delle funzioni. Si è previsto un tempo approssimativo di 5 settimane di lavoro per il completamento del codice più ulteriori 3 settimane per apportare eventuali modifiche perfettive, nell'ipotesi che in prima compilazione alcune richieste non siano state soddisfatte. Non vengono conteggiate all'interno di questo progetto eventuali altre modifiche o correzioni che potranno pervenire oltre questo lasso di tempo.

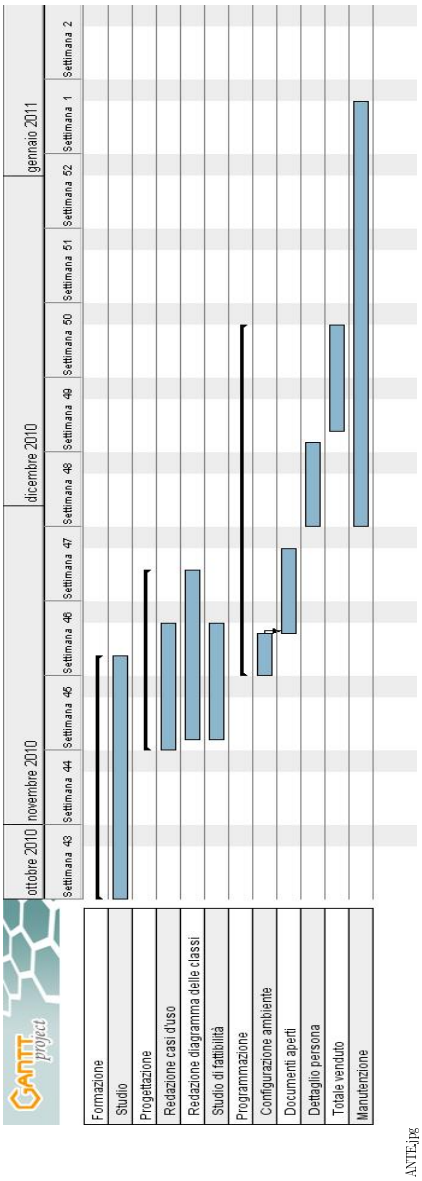

Figura 2.3: Diagramma di Gantt modulo svendite

Stima dei costi

Oltre il tempo il project manager deve saper stimare anche i costi di un progetto. Per la stima dei costi esistono varie tecniche. Si possono conteggiare le linee di codice e applicare un costo ad ogni linea scritta oppure

utilizzare i function point che misurano la complessità dei requisiti funzionali del progetto [20]. I function point si dividono in categorie:

- External Input: informazioni e i dati provenienti dall'utente
- External output: informazioni e dati forniti all'utente
- External Inquiry: sequenze interattive di richieste-risposte
- External Interface File: interfaccie con altri sistemi informativi esterni
- Internal Logical File: file principali logici gestiti nel sistema. [6]

Per stimare lo sforzo di ogni function point si è utilizzato il modello COCOMO. COCOMO è un modello matematico utilizzato per la valutazione dei function point, il modello consente l'inserimento dei function point divisi in 3 classi di difficoltà. Inserito questo parametro principale è possibile inserire dei moltiplicatori per aumentare l'accuratezza della valutazione effettuata. I moltiplicatori sono 6:

- PERS: capacità del personale
- RCPX: complessità del prodotto
- RUSE: livello di riuso
- PDIF: difficoltà legate alla piattaforma utilizzata
- PREX: esperienza del personale
- FCIL: funzionalità di supporto
- SCED: tempistica

Ad ognuno si può applicare il grado di complessità relativo al modulo da sviluppare. [8] Di seguito presento il report della stima dello sforzo per il progetto che andrò a sviluppare.

Per questo progetto COCOMO prevede uno sforzo di 6.5 mesi persona, oppure ottimisticamente 4.4 mesi persona. Viene inoltre riportato il costo

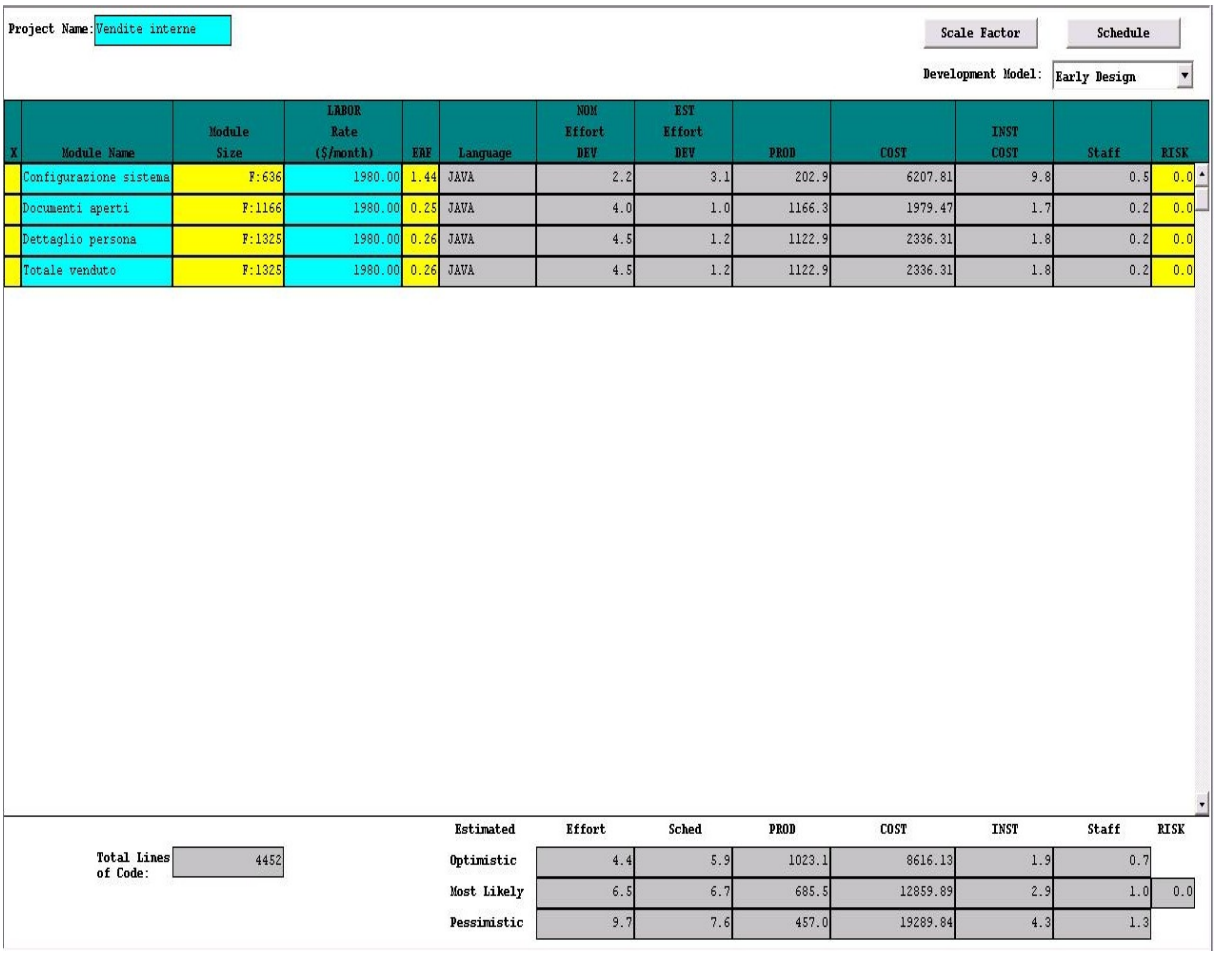

Figura 2.4: Stima dello sforzo

del prodotto che si aggira tra gli 8600 euro e i 12800 euro. Il modulo più complesso è la configurazione del sistema; benchè sia il più ridotto a livello di codice da redarre, è quello che richiede lo sforzo maggiore, siccome è gravato dalla inesperienza del personale che si ritrova per la prima volta ad operare con Struts 2 e a doverlo innestare nel progetto esistente. Quindi all'interno di questo modulo sono presenti anche i costi della formazione del personale. I restanti tre moduli non sono particolarmente complicati in quanto sono di ordinaria routine per il team di sviluppo. Prevedono tutti e tre la creazione di un file excel quindi hanno un alto tasso di riusabilità. Come sopra, l'unica vera difficoltà è il dover cambiare approccio di configurazione per la chiamata

delle azioni, quindi il primo modulo che verrà sviluppato sarà soggetto a problemi di bassa produttività siccome è il primo frammento sviluppato con la nuova tecnologia e quindi dovrà essere testato più a fondo degli altri due.

2.3 Progettazione

Il modulo che si è sviluppato è progettato per offrire all'utente uno strumento semplice ed intuitivo che gli permetta di estrarre i dati in piena autonomia. Il layout grafico dovrà essere conforme al precedente layout in modo che l'utente non noti la discontinuità di implementazione tecnica e soprattutto non si ritrovi disorientato dalla nuova veste grafica. Nella figura 2.5 è riportato il diagramma ER su cui poggia il progetto. Nella figura 2.6 è presente il diagramma delle classi del modulo sviluppato. Le classi da implementare prevedono prima di tutto la creazione attraverso Hibernate della mappatura del database di modo che si può operare su ogni record come fosse un oggetto. Dopodichè si creeranno le classi sulle quali poggiano le logiche di business. Per ogni funzione è presente una classe che incarna l'azione, suffisso Action, e una classe utilizzata per interfacciarsi con il database, suffisso Database. Ogni classe Action implementerà i metodi:

- cerca(), che sarà richiamato dal punto di menù e restituirà il form di ricerca
- applicaCerca(), che verrà richiamato dal form di ricerca e avrà come risultato il file xls
- salvaExcel() metodo interno alla classe che viene utilizzato per scorporare la creazione del file xls dalla logica della Action

Il metodo che incarna la logica della funzione è applicaCerca(), che prima di tutto deve controllare la correttezza degli input, successivamente demanda all'oggetto Database la creazione dell'interrogazione al database e infine crea il file xls con la risposta dal database.

2.4 Implementazione

Il progetto sviluppato è una web application e come tale richiede l'implementazione di alcuni file di configurazione indispensabili. Questi file devono essere i primi ad essere implementati, siccome sono il cuore della web app e sono richiesti in fase di start up di quest'ultima per il suo corretto caricamento e funzionamento. Il nome di questi file sono:

- web.xml
- struts.xml.

Nel primo file sono elencate tutte le componenti fondamentali della nostra applicazione come le Servlet utilizzate o i filtri creati.

```
<filter>
      <filter-name>Struts2</filter-name>
      <filter-class>
        org.apache.Struts2.dispatcher.FilterDispatcher
      </filter-class>
</filter>
<filter-mapping>
     <filter-name>Struts2</filter-name>
     <url-pattern>*.action</url-pattern>
</filter-mapping>
```

Listato 2.1 Frammento web.xml

Il codice riportato nel listato 2.1 mostra la creazione del filtro standard di Struts 2 chiamato FilterDispatcher che si farà carico di instradare le richieste provenienti dall'utente. Dichiarando la clausola <filter-mapping> si è fatto in modo che FilterDispatcher risponda solo a richieste con estensione .action. Questo accorgimento è reso necessario dal fatto che la nostra applicazione lavora con due framework differenti che usano strutture di controllo differenti

tra di loro, quindi deve saper distinguere quale tipo di controller utilizzare se la Servlet di Struts o il Filter di Struts 2.
Nel secondo file invece sono mappate tutte le azioni di Struts 2. Rispetto a Struts gli è stata cambiata la locazione che non è più sotto WEB-INF, ma bensì sotto src quindi insieme alle classi .Java.

```
<package name="/coge/svendite/dettaglioPersona" extends="Struts-default">
<result-types>
      <result-type name="tiles"
                  class="org.apache.Struts2.views.tiles.TilesResult"/>
</result-types>
<action name="cerca" method="cerca"
class="it.bc.coge.svendite.action.DettaglioPersonaAction">
    <result name="SUCCESS" type="tiles">/dettaglioPersona.tiles</result>
</action>
<action name="applicaCerca" method="applicaCerca"
class="it.bc.coge.svendite.action.DettaglioPersonaAction">
    <result name="SUCCESS" type="tiles">/dettaglioPersona.tiles</result>
</action>
</package>
```

Listato 2.2 Frammento struts.xml

Il file struts.xml come detto ha la funzione di mappare le azioni. Il listato 2.2 mostra un esempio della mappatura delle azioni della funzione DettaglioPersonaAction. I metodi mappati sono due:

- cerca che viene richiamato dal punto di menù e serve a mostrare la jsp di ricerca
- applicaCerca che viene richiamato dal pulsante Invia della jsp di ricerca, che fa partire l'interrogazione al database.

Fatte queste piccole modifiche ai file di configurazione, ci possiamo dedicare all'implementazione delle classi Java. Per prima cosa con l'ausilio di Hiber-

nate [7] si creano le classi di servizio. Facendo questo si ha la possibilità di trattare ogni record del nostro database come se fosse un oggetto. Il grande vantaggio è che Hibernate importa anche le relazione tra gli oggetti e ci permettere di creare interrogazioni al database che ci restituiscono una lista di oggetti sui quali poi possiamo operare con maggiore facilità. Un esempio di mappatura di una tabella con l'ausilio di Hibernate è mostrato nel listato 2.3.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "
-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="it.bc.hibernate.tabelle.svendite.Baaesrr00f"
       table="BAAESRR00F" schema="KSIADATICG">
        <comment>Svendita reso riga</comment>
        <id name="baaeId" type="it.bc.hibernate.dao.TrimmedStringType">
            <column name="BAAE_ID" length="13" />
            <generator class="assigned" />
        </id>
        <many-to-one name="baadsrt00f"
          class="it.bc.hibernate.tabelle.svendite.Baadsrt00f"
           fetch="select" insert="false" update="false">
            <column name="BAAE_ID_BAADSRT00F" length="13" not-null="false">
                <comment>ID_BAADSRT00F</comment>
            </column>
        </many-to-one>
        <many-to-one name="baacscr00f"
          class="it.bc.hibernate.tabelle.svendite.Baacscr00f"
          fetch="select" insert="false" update="false">
            <column name="BAAE_ID_BAACSCR00F" length="13" not-null="false">
                <comment>ID_BAACSCR00F</comment>
```

```
            </column>
        </many-to-one>
        <property name="baaeAziendaCodice"
         type="it.bc.hibernate.dao.TrimmedStringType">
            <column name="BAAE_AZIENDA_CODICE" length="5" not-null="true">
                <comment>AZIENDA CODICE</comment>
            </column>
        </property>
        <property name="baaeIdBaadsrt00f"
          type="it.bc.hibernate.dao.TrimmedStringType">
            <column name="BAAE_ID_BAADSRT00F" length="13" not-null="true">
                <comment>ID BAADSRT00F</comment>
            </column>
        </property>
        <property name="baaeIdBaacscr00f"
           type="it.bc.hibernate.dao.TrimmedStringType">
            <column name="BAAE_ID_BAACSCR00F" length="13"
               not-null="true" unique="true">
                <comment>ID BAACSCR00F</comment>
            </column>
        </property>
        <property name="baaeCreazioneData" type="date">
            <column name="BAAE_CREAZIONE_DATA" length="10" not-null="true">
                <comment>CREAZ.DATA</comment>
            </column>
        </property>
    </class>
</hibernate-mapping>
```

Listato 2.3 Baaesrr00f.hbm.xml

Svolte queste operazioni preliminari, ma necessarie al corretto funzionamento del nostro applicativo ci possiamo dedicare alla implementazione delle funzioni richieste dall'azienda. Come detto le funzioni richieste sono tre, quindi per ognuna creeremo la classe Action e la classe Database, questa distinzione nasce per aiutare la futura manutenzione, perchè divide la logica di Struts con le chiamate al database, e per tenere più pulito il nostro codice. Il metodo più importante è applicaCerca, perchè è da questo metodo che partono le richieste al database. Grazie a Struts 2 non è più necessario creare un form siccome i dati presenti nella jsp di ricerca vengono caricati in automatico nelle variabili interne alla classe Action.

Dettaglio per persona

Questa funzione nasce per soddisfare il bisogno di ricercare lo stato di una persona rispetto ad uno o più eventi ricercati. L'azienda ha richiesto la possibilità di ricercare in uno specifico range di date per uno o più eventi. Ha inoltre richiesto che l'inserimento della persona da ricercare sia facoltativo quindi che si possa anche analizzare lo stato di uno o più eventi nel loro complesso. Per facilitare l'uso della funzione all'interno della maschera jsp è stato inserita una funzione di autocompletamento, vedi listato 2.4, del nome o cognome della persona che tramite la digitazione di minimo 4 caratteri compie una ricerca all'interno del database e restituisce tutte le persone che hanno all'interno del nome o del cognome la stringa cercata.

```
<ajax:autocomplete
 parameters="finestra=${requestScope.userContainer.id},
 funzione=idPersonaInterna"
 postFunction="writeAjaxClienteDa"
 source="ajaxPersonaInterna"
 target="persona"
 baseUrl="${path}autocomplete.view"
 className="autocomplete"
 indicator="indicator5"
```

```
 minimumCharacters="4"
/>
```

Listato 2.4 Codice in jsp per autocompletazione

L'azienda ha richiesto di poter visualizzare il dettaglio di ogni singola transazione avvenuta all'interno dei criteri di ricerca effettuati, quindi vuole sapere chi è stato ad effettuare la transazione e a quale categoria appartiene, quando è stata effettuata, il modello, il numero del documento, il prezzo e il costo industriale del modello e lo stato del documento. Per effettuare la richiesta al database si è usato Hibernate che attraverso i suoi metodi permette in modo semplice e pulito la creazione di query complesse e la restituzione di oggetti facilmente utilizzabili. Il listato 2.5 è un piccolo esempio di come adoperare Hibernate e Criteria per la creazione di una interrogazione al database.

```
Criteria criteria = session.createCriteria(Baaesrr00f.class);
criteria = criteria.createAlias("baacscr00f.baabsct00f", "baabsct00f",
                                                    Criteria.INNER_JOIN);
criteria = criteria.add(Restrictions.eq("baadsrt00f.baadResoVendita", "V"));
criteria.addOrder(Order.asc("baadsrt00f.baadCreazioneData"));
List<Baaesrr00f> lista = criteria.list();
```

Listato 2.5 Esempio creazione query con criteria

Per la creazione del file xls si sono utilizzate le librerie HSSFPOI. Queste librerie sono un semplice e valido strumento che ci permetta la genesi di file xls di report. Il listato 2.6 mostra le istruzioni base necessarie per scrivere un foglio xls.

```
HSSFWorkbook dati = new HSSFWorkbook();
HSSFSheet foglio = dati.createSheet("Dati");
Row row = foglio.createRow(0);
row.createCell(0).setCellValue("Data emissione");
OutputStream out = response.getOutputStream();
response.setContentType("application/octet-stream");
```

```
response.setHeader("content-disposition",
"attachment;filename=\"DettaglioSvendite.xls\"");
dati.write(out);
out.close();
```

Listato 2.6 Esempio creazione foglio xls con HSSFPOI

Per soddisfare il requisito di sicurezza che l'azienda ha richiesto, cioè la possibilità che solo alcuni utenti autorizzati possano accedere ai dati, si è utilizzato una funzione già presente all'interno del ERP. Questa funzione prevede la chiamata ad una tabella della autorizzazioni in cui sono salvati gli username dei dipendenti e le relative autorizzazioni. Se l'utente non è presente restituisce una pagina standard in cui si nota la mancanza dell'autorizzazione. Questo controllo viene effettuato al momento del lancio di applicaCerca(). Si è deciso di permettere a tutti gli utenti di accedere al form di ricerca della funzione in modo che gli utenti interessati possano vedere cosa c'è sotto il punto di menù, ma si è deciso di proteggere i dati quindi anche se l'utente può accedere al form non può effettuare alcuna ricerca. Quindi gli utenti interessati dovranno fare richiesta della autorizzazione necessaria per accedere ai dati. In figura 2.7 viene mostrata la pagina jsp di ricerca per la funzione dettaglio per persona. Sono presenti dei campi per le date con apposito calendario per l'inserimento di un range temporale, un campo di autocompletamento per la ricerca di una persona e una lista a scelta multipla implementata con l'utilizzo di JQuery. La veste grafica è ereditata delle precedenti funzioni sviluppate per questo ERP.

Documenti aperti

Questa funzione ha il compito di mostrare in dettaglio tutti i documenti ancora non pagati. Si ha la possibilità di limitare la ricerca a uno specificato evento o ad un range ristretto di date oppure ad una singola categoria di persone. Di ogni documento aperto si vuole sapere il numero del documento, chi ha acquisito il capo, che modello, quando è stato emesso il documento e il prezzo del capo.

Totale venduto

Il compito di questa funzione è la ricerca del totale venduto. La specifica richiede di sapere un totale riassuntivo di tutti i capi venduti. Si vuole poter raggruppare questi capi per giorno o per tipo filato e si vuole sapere anche il dettaglio dei capi venduti. L'azienda ha richiesto di poter ricercare uno o più eventi, di poter cercare in un numero ristretto di date, di scegliere il tipo di raggruppamento e di poter limitare la ricerca ad una sola linea di vendita. Siccome l'azienda ha richiesto di visualizzare i dati sia riassuntivi che in dettaglio si è scelto di mostrare un file xls diviso in più fogli. Nel primo foglio saranno presenti i dati riassuntivi come il numero totale dei capi venduti, il totale incassato, il costo industriale complessivo,la data o il tipo filato e l'evento, raggruppati o per giorno o per tipo filato. Il file xls poi, conterrà un foglio per ogni giorno o per ogni tipo filato presente e riporterà all'interno i dati in dettaglio di ogni singolo capo venduto.

Software utilizzato

Per lo sviluppo di questa tesi sono stati utilizzati i seguenti supporti software:

- Eclipse IDE per lo sviluppo del codice
- SQuirreL SQL per la creazione di query SQL e monitoraggio del database
- ModelSphere per la creazione del modulo E/R
- GanttProject per la creazione del diagramma di Gantt
- Xmind per la creazione del WBS
- COCOMO II.2000.0 software per la stima dei costi tramite il metodo COCOMO
- ArgoUML per la creazione dei report UML

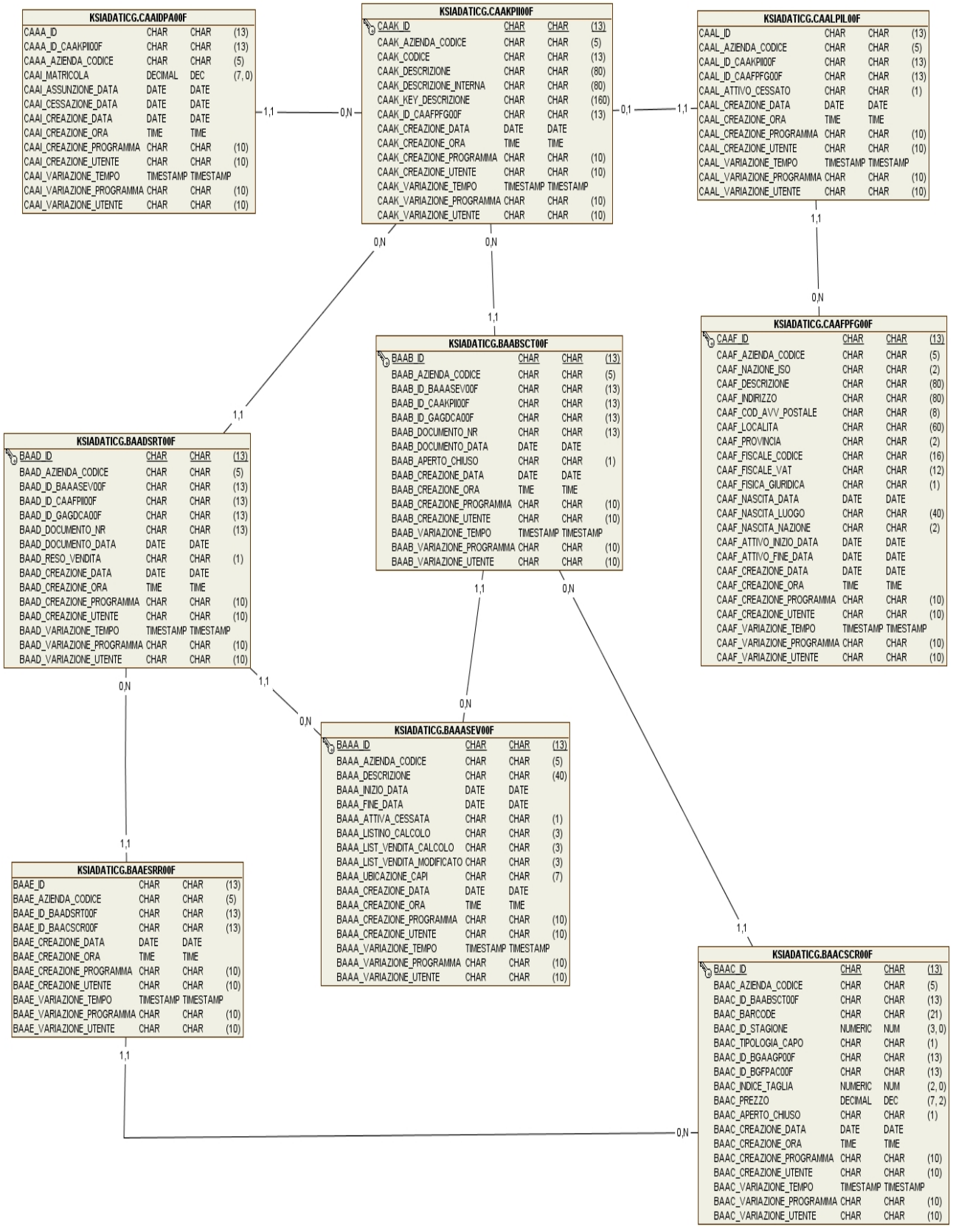

Figura 2.5: Diagramma ER modulo svendite

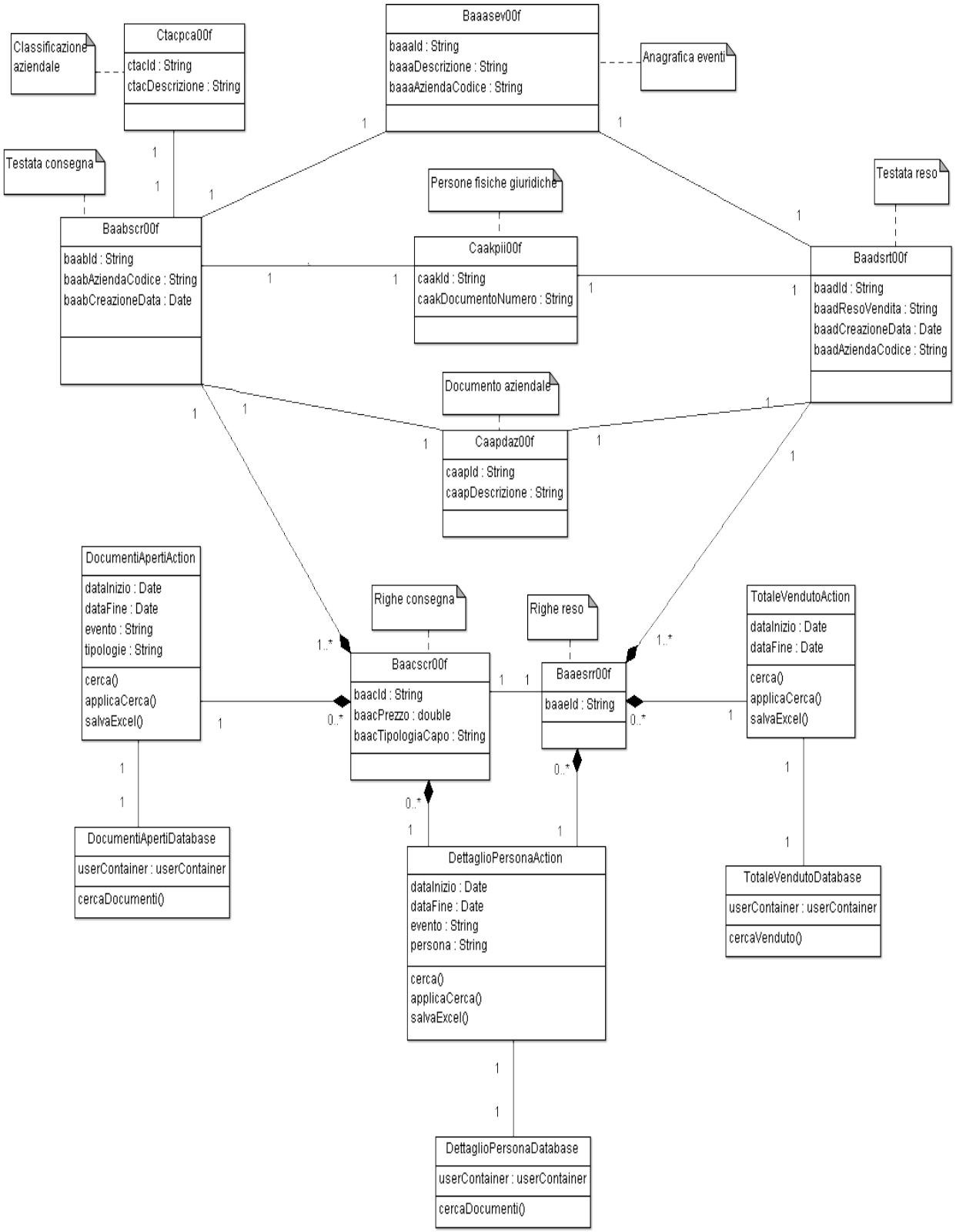

Figura 2.6: Diagramma delle classi modulo svendite

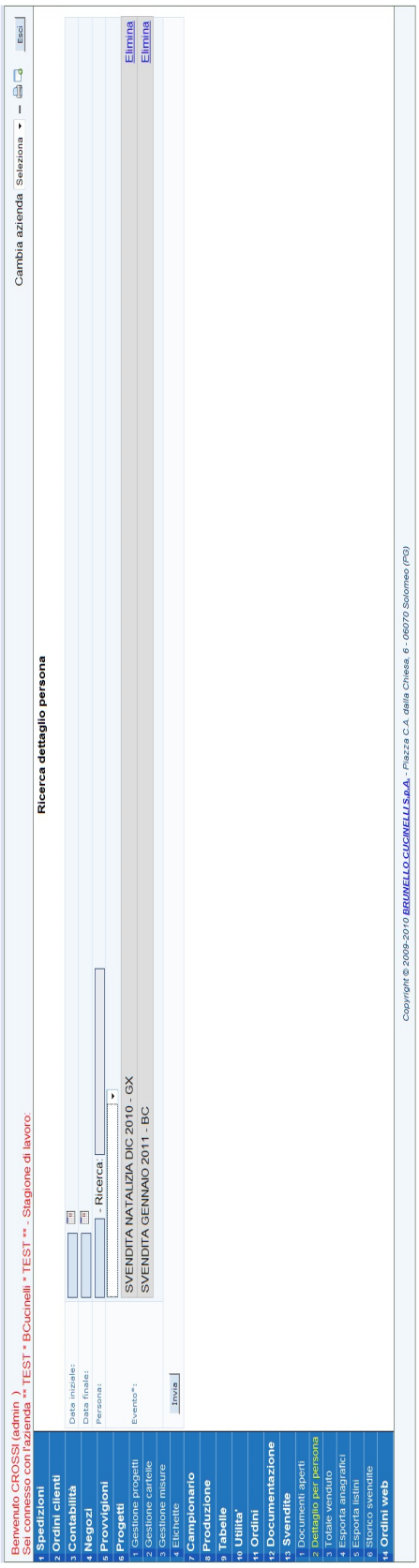

Figura 2.7: Form di ricerca dettaglio per persona

Capitolo 3

Misura delle prestazioni

Pochi giorni dopo il rilascio della prima release del modulo l'azienda ha richiesto una modifica nella funzione che estrae il dettaglio per persona. Hanno richiesto di poter estrarre il file senza dover selezionare una persona specifica, cioè di poter vedere il dettaglio delle svendite selezionate per intero.
Eseguita questa modifica sono sorti seri problemi prestazionali. La funzione è passata da pochi secondi di calcolo a molti minuti. Alcune avvisaglie di questo problema erano note siccome se si selezionava un utente con molti capi a carico il tempo di esecuzione si dilatava, ma non si è indagato siccome erano casi isolati e comunque i tempi rimanevano accettabili sotto i 45 secondi. Con l'apporto di questa modifica i tempi di computazione per la ricerca del dettaglio di una svendita si aggirava intorno ai 6/7 minuti con punte anche di 10/12 minuti a seconda dello stato di stress della macchina. I nuovi tempi erano, giustamente, inaccettabili quindi vi si è dovuto porre una soluzione.
Indagando i file di log restituiti da Hibernate si è evidenziata la causa del problema. Si è scoperto che le mappature Hibernate avevano impostato il caricamento lazy dei dati dalle tabelle in join, questo comportamento consente ad Hibernate di caricare al momento dell'esecuzione solo la tabella di from e le tabelle a lei più vicine, mentre i campi delle tabelle più lontane alla tabella di partenza vengono ricercati solo nel momento in cui vengono

richiesti. Questo sistema consentirebbe un notevole risparmio di risorse nel caso si avesse bisogno solo sporadicamente delle informazioni presenti nella tabelle più lontane, ma nel caso presentato quelle informazioni servivano per ogni riga estratta; questo comportava l'effettuazione di due interrogazioni per ogni riga e quindi un serio aggravio nei tempi di esecuzione.
Come prima soluzione si è pensato di eliminare il caricamento in modalità lazy, perchè avrebbe consentito una modifica rapida e pulita del codice. Questo sistema durante i test si è rilevato inefficiente: i tempi non erano migliorati affatto anzi nei primi test i tempi di esecuzione erano più lunghi dei precedenti.
Alla luce di questi fatti si è, quindi, deciso di modificare il codice e passare all'implementazione della query con HQL query [12]. Questa modifica ha riportato l'applicazione a tempi di utilizzo accettabili riportando i tempi medi di esecuzione sotto il minuto con punte di massimo di un minuto e mezzo nel caso la macchina sia particolarmente stressata.

Conclusioni

Lo studio ha rilevato che la sostituzione ha comportato oneri al di sopra delle aspettative.
Il nuovo framework ha grandi potenzialità, ma rispetto al suo predecessore le tecniche di base sono state stravolte quindi passare rapidamente a Struts 2 per un programmatore di Struts risulta difficoltoso.
Durante l'implementazione si è riscontrata particolari difficoltà nel far lavorare insieme i due framework in particolar modo la veste grafica e il richiamo di funzioni già sviluppate in precedenza. Infatti il nuovo framework ha preso molto più da WebWork che da Struts e quindi il passaggio risulta difficoltoso con una curva di apprendimento lunga e ripida.
In figura 3.1 è riportato il diagramma di Gantt a consuntivo del progetto. Rispetto a quanto previsto i tempi di consegna del progetto si sono notevolmente dilatati, portando così anche un aggravio dei costi di realizzo. I costi di progetto sono aumentati di quasi il 100% rispetto ai costi previsti in fase di progettazione. Questo aumento è dovuto in particolar modo perchè, si sono dilatati i tempi di programmazione del codice che sono stati doppi rispetto a quelli pronosticati e quindi il costo della manodopera è raddoppiato. Questo è dovuto al fatto che ci si è imbattuti nel problema che affligge alcuni framework open source, cioè la mancanza di documentazione. Questo problema nasce se non esiste una community sviluppata e attiva di sviluppatori e soprattutto di utilizzatori del framework che riscontrano problemi e postano soluzioni e consigli per i nuovi utenti.
Il progetto, quindi, benchè realizzabile ha costi troppo alti per benefici ridot-

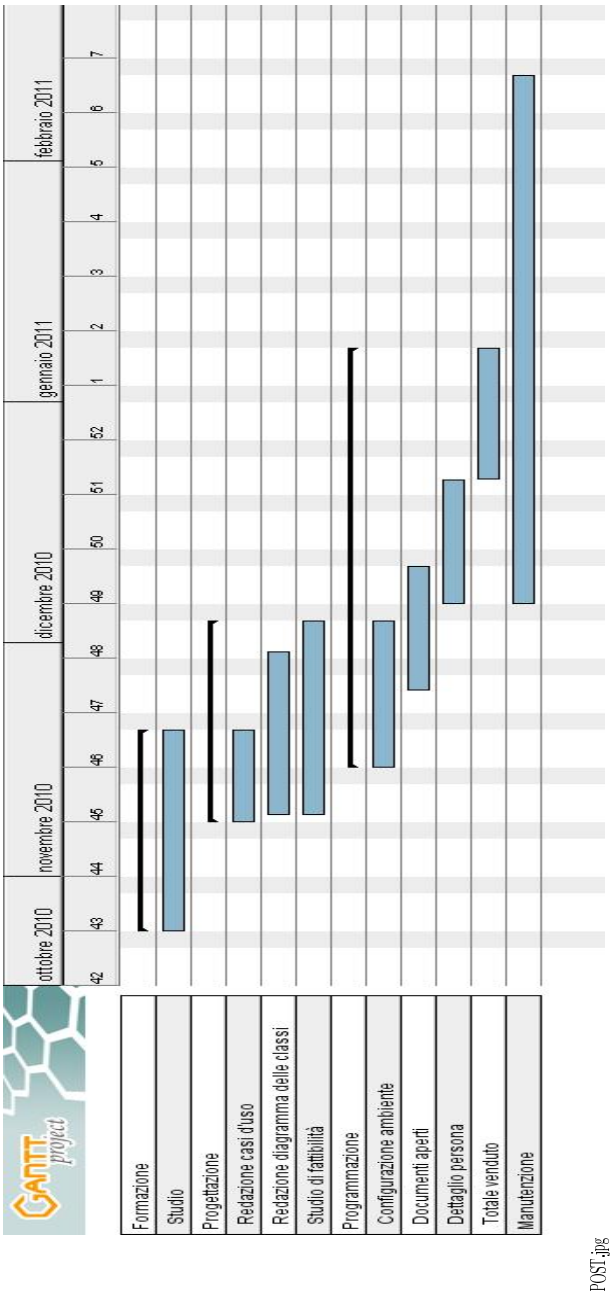

Figura 3.1: Diagramma di Gantt a consuntivo

ti.

In conclusione alla luce di quanto studiato e realizzato per la creazione di questo elaboarto, posso sostenere che Struts 2 sia un ottimo framework. In-

tegra molte nuove funzioni ed è ben costruito, ma non soddisfa i requisiti che avevamo richiesto come successore tecnologico di Struts. I requisiti erano:
la buona integrazione con il progetto sviluppato in precedenza, requisito non soddisfatto in pieno a causa di particolari problemi riscontrati nella realizzazione della veste grafica e del richiamo di funzioni realizzate in precedenza;
costi di apprendimento e sviluppo limitati, anche questo requisito non è stato soddisfatto in pieno siccome i costi e i tempi di realizzo sono raddoppiati rispetto a quelli previsti in precedenza.
Alla luce di questi fatti, Struts 2 non si è rilevato una piattaforma ideale per la successione di Struts come framework implementativo per la realizzazione di nuovi moduli in seno al progetto esistente; questa scelta deriva dagli ingenti costi di apprendimento e di affinamento delle tecniche di programmazione, che possono essere gestiti come costi fissi da ammortizzare nel tempo, ma che non sono sufficientemente ammortizzati durante la realizzazione di piccoli moduli. Questi costi, invece potrebbero essere ben ammortizzati durante la realizzazione di un progetto di dimensioni più sostenute.

Bibliografia

[1] Struts 2 vs struts 1. http://www.struts2.net/comparison.htm. [Online; accessed 07-November-2010].

[2] AIRI. Andamento della ricerca e sviluppo in italia e nei principali paesi. http://progetti.airi.it/statistiche-ricerca-sviluppo/, 2010. [Online; accessed 26-October-2010].

[3] Luciano Alessandro Ipsaro Palesi Andrea Marzilli, Simone Pascuzzi. Guida apache struts. http://java.html.it/guide/leggi/181/guida-apache-struts/. [Online; accessed 03-November-2010].

[4] Luciano Alessandro Ipsaro Palesi Andrea Marzilli, Simone Pascuzzi. Guida apache struts. http://java.html.it/guide/lezione/4555/componenti-e-gestione-delle-richieste-in-struts/. [Online; accessed 03-November-2010].

[5] Claudio De Sio Cesari. Mvc. http://www.claudiodesio.com/ooa&d/mvc.htm. [Online; accessed 21-February-2011].

[6] Paolo Ciancarini. Project management dei progetti software. Technical report, Università di Bologna, 2010.

[7] JBoss Community. Hibernate. http://www.hibernate.org/, 2011. [Online; accessed 21-February-2011].

[8] Anna Rita Fasolino. Il modello cocomo per la stima dei costi software la gestione dei rischi. Technical report, Università Federico II di Napoli, 2009.

[9] Apache Software Foundation. Apache struts. `http://struts.apache.org/1.3.10/index.html`, 2008. [Online; accessed 21-February-2011].

[10] Apache Software Foundation. Apache struts 2. `http://struts.apache.org/2.2.1/index.html`, 2010. [Online; accessed 21-February-2011].

[11] Apache Software Foundation. Struts 2 request flow. `http://struts.apache.org/2.0.14/docs/the-struts-2-request-flow.html`, 2011. [Online; accessed 21-February-2011].

[12] Inc. Red Hat. Hql query. `http://docs.jboss.org/hibernate/core/3.3/reference/en/html/queryhql.html`, 2007. [Online; accessed 21-February-2011].

[13] Ian Roughley. *Starting Struts 2.* InfoQ.com, 2006.

[14] Ian Roughley. *Practical Apache Struts2 Web 2.0 Project.* Apress, 9th edition, 2007.

[15] WebSiteOptimizer.com. Ajax. `http://www.websiteoptimization.com/secrets/ajax/8-1-ajax-pattern.html`, 2009. [Online; accessed 21-February-2011].

[16] Wikipedia. Java applet. `http://it.wikipedia.org/wiki/Java_applet`, 2009. [Online; accessed 27-October-2010].

[17] Wikipedia. Ajax. `http://it.wikipedia.org/wiki/AJAX`, 2010. [Online; accessed 02-November-2010].

[18] Wikipedia. Apache strtus. `http://it.wikipedia.org/wiki/Apache_Struts`, 2010. [Online; accessed 03-November-2010].

[19] Wikipedia. Cgi. http://it.wikipedia.org/wiki/Common_Gateway_Interface, 2010. [Online; accessed 27-October-2010].

[20] Wikipedia. Function point. http://it.wikipedia.org/wiki/Function_point, 2010. [Online; accessed 27-November-2010].

[21] Wikipedia. Java servlet. http://en.wikipedia.org/wiki/Java_Servlet, 2010. [Online; accessed 30-October-2010].

[22] Wikipedia. Javascript. http://it.wikipedia.org/wiki/Javascript, 2010. [Online; accessed 29-October-2010].

[23] Wikipedia. Mvc. http://it.wikipedia.org/wiki/Model-View-Controller, 2010. [Online; accessed 02-November-2010].

[24] Wikipedia. Web application framework. http://it.wikipedia.org/wiki/Web_Application_framework, 2010. [Online; accessed 02-November-2010].

[25] Wikipedia. Cocomo. http://it.wikipedia.org/wiki/COCOMO, 2011. [Online; accessed 21-February-2011].

[26] Wikipedia. Diagramma di gantt. http://it.wikipedia.org/wiki/Diagramma_di_Gantt, 2011. [Online; accessed 21-February-2011].

[27] Wikipedia. Erp. http://it.wikipedia.org/wiki/Enterprise_Resource_Planning, 2011. [Online; accessed 21-February-2011].

[28] Wikipedia. Esternalizzazione. http://it.wikipedia.org/wiki/Esternalizzazione, 2011. [Online; accessed 21-February-2011].

[29] Wikipedia. Wbs. http://it.wikipedia.org/wiki/Work_Breakdown_Structure, 2011. [Online; accessed 21-February-2011].

Ringraziamenti

Vorrei ringraziare tutte le persone che mi hanno permesso di raggiungere questo importante traguardo. In particolar modo ringrazio il Chiar.mo Prof. Antonio Messina, che mi ha seguito nella redazione di questo elaborato. Ringrazio anche i miei genitori che si sono sacrificati per permettermi di studiare. Vorrei inoltra ringraziare il mio collega l'Ing. Riccardo Camanzi per il suo ruolo di mentore nella mia vita lavorativa. Ultimi, ma non per importanza i miei compagni di corso che mi sono stati vicini in questi 3 anni lontano dalla mia terra natia.