UNIVERSITÀ DEGLI STUDI DI MODENA
E REGGIO EMILIA

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea in Ingegneria Informatica

Portale Web di Facoltà: progetto e implementazione della funzione di gestione di informazioni e materiale didattico a cura del docente

Relatore

Chiar.mo Prof.

Sonia Bergamaschi

Correlatore

Ing. Maurizio Vincini

Tesi di Laurea di

Luca Ciocci

Anno Accademico 2000-2001

Ringraziamenti

Desidero ringraziare la Professoressa Sonia Bergamaschi per l’aiuto fornito durante la realizzazione della tesi e per la disponibilità dimostrata.
Un ringraziamento anche all’Ing. Maurizio Vincini e l’Ing. Francesco Guerra per la collaborazione e l’interesse dimostrato nello svolgimento di questa tesi di laurea.
Un rigraziamento va anche dato allo staff della ASK s.n.c per avermi permesso di utilizzare le apparecchiature e il software a disposizione della società per agevolarmi nello sviluppo della applicazione web, e per aver tollerato le mie saltuarie assenze dal posto di lavoro.

Infine desidero ringraziare la mia fidanzata Luisa per la pazienza dimostrata in questi sei mesi.

Indice

Elenco Figure

Elenco Tabelle

Capitolo 1
Specifiche dei requisiti

Questo argomento è trattato in parte anche negli altri progetti relativi al portale web, tuttavia in questa circostanza viene eseguita una analisi del problema in modo più approfondito. Inoltre essendo questa la continuazione di progetti precedenti si devono per ragion di logica, effettuare alcune considerazione sulle scelte tecnologiche utilizzate precedentemente.

1.1 Considerazioni Generali

Come qualsiasi applicazione software anche le applicazioni web necessitano di abbondanti specifiche dei requisiti per essere modellate in modo efficiente. Le specifiche dei requisiti si dividono in specifiche funzionali e non funzionali. Le prime sono le più interessanti perché descrivono da parte del committente ciò che deve svolgere l'applicazione. Le seconde invece vengono tipicamente scelte dalla software house in base alle esperienze dei propri tecnici e programmatori, oppure, in base ad esperienze raccolte in progetti precedenti e alla tipologia del progetto.

Considerando l'ambito universitario, e che il progetto del portale web di facoltà è in corso di sviluppo, tutte le specifiche non funzionali sono già state fissate in base alle risorse messe a disposizione dell'Università, e comunque data questa particolare circostanza più che di specifiche non funzionali si dovrebbe parlare di vincoli progettuali, in quanto il progetto ha già preso forma ed è attualmente in fase di sviluppo.

Per tutto ciò che riguarda invece le specifiche funzionali, è possibile fare riferimento anche ai documenti precedenti, nel caso in cui si voglia avere una visione globale del progetto. Alcune linee generali comunque verranno riportate di seguito.

Il portale web di facoltà rappresenta senza dubbio un applicazione web molto complessa, sia per la vastità del progetto, che per le particolarità dell'ambiente universitario. Basti pensare solamente alla gestione delle informazioni riguardanti la didattica per valutare la rapidità e la flessibilità necessarie per una applicazione web che deve rappresentare i contenuti di questa sezione del portale. Il sito

web deve dunque essere concepito con una struttura molto flessibile ed aperta, in grado di adattarsi ad eventuali variazioni future. L'aggiornamento frequente delle informazioni, dovuto a questa situazione in continuo mutamento, comporta l'esigenza di concepire funzioni di aggiornamento dei contenuti del portale, rapide e di facile utilizzo. Come qualsiasi altra applicazione, il portale web di facoltà deriverà il suo successo in parte anche alla facilità di utilizzo, sia da parte di coloro che navigano il sito alla ricerca di informazioni, sia per coloro che mantengono aggiornate queste informazioni. Per questo motivo l'accesso alla area di amministrazione del sito web, dovrà essere in grado di discriminare la tipologia dell'utente che ha avuto acceso. Ogni tipologia di utente poi avrà la possibilità di gestire solamente le informazioni di sua competenza.

In primo luogo è necessario dunque individuare quali sono i tipi di utenti che hanno accesso al sito:

- Studenti
- Docenti
- Visitatori
- Preside e altre cariche istituzionali

Con riferimento alla realizzazione di questa parte del progetto particolarmente importanti risultano i docenti e gli studenti.
Tuttavia i requisiti sono stati divisi nelle seguenti tipologie:

- Requisiti relativi agli aspetti istituzionali
- Requisiti relativi alla didattica
- Requisiti relativi ai servizi collaterali per gli studenti

L'applicazione realizzata dovrà dunque tenere in considerazione in particolare modo dei requisiti relativi agli aspetti istituzionali e cercare di non violarli.

1.2 Vincoli Progettuali

Per sviluppare in modo efficiente un portale web come quello della facoltà di Ingegneria Informatica è necessario scegliere tecnologie largamente diffuse per cui esista una ottima documentazione, ma soprattutto la tecnologia scelta deve essere aperta e affidabile, deve cioè essere in grado di adattarsi ad eventuali nuove esigenze.

Realizzare una applicazione web di questo tipo pone dei vincoli inviolabili per ciò che riguarda requisiti software minimi. Sono necessari infatti i seguenti pacchetti:

- ?? **Web Server:** per la presentazione di tutte le pagine statiche
- ?? **Application Server :** per la presentazione delle pagine dinamiche
- ?? **RDBMS:** per la memorizzazione in modo efficiente ed efficace di grandi quantità di informazioni

Essendo già stata definita la raccolta dei requisiti, e la realizzazione di una analisi di massima della applicazione, è evidente che siano già state fatte le scelte relative alle specifiche non funzionali, per tale motivo in questo contesto si parlerà di **vincoli progettuali** e comunque verrà analizzata la bontà delle scelte effettuate in precedenza. Questi vincoli consistono in particolare nelle risorse hardware e software messe a disposizione dalla Università di Modena e Reggio Emilia, e non possono essere violate dalla applicazione web che sarà realizzata.

Per primo il linguaggio di programmazione scelto è Java 2 della Sun Microsystem, una piattaforma moderna e perfettamente adatta allo sviluppo di applicazioni web aperte e flessibili e di prestazioni soddisfacenti. Inoltre guardando al futuro, sarà presto disponibile una versione di Java detta Enterprise Edition pensata per una connettività globale (consente infatti di interfacciarsi a server di posta, server LDAP ecc, vedere appendice 2 per maggiori informazioni). Quindi per quanto riguarda le caratteristiche di espandibilità di Java, non vi è alcun dubbio che la scelta fatta sia tra le migliori. Infatti attualmente questo linguaggio di programmazione è diffusissimo per la realizzazione di siti web dinamici.

In secondo luogo il sistema operativo sul quale si appoggia l'applicazione è un Unix (o Unix like), in quanto fornisce maggiore sicurezza di protezione dei dati e più controllo sugli accessi degli utenti.

Il web server utilizzato sarà Apache, che gira sulla totalità dei sistemi operativi Unix ed è noto per la sua robustezza e affidabilità. Come conseguenza di ciò l'Application Server deve essere facilmente interfacciabile ad Apache, fortunatamente ne esistono diversi tipi in commercio, ma la scelta più opportuna per questo progetto è quella di utilizzare Tomcat del progetto Jakarta, una sezione della Apache Software Foundation.

Una nota particolare va detta per quanto riguarda il Relational Database Management System (RDBMS) scelto per memorizzare i dati. In fase di test e di sviluppo si è utilizzato Interbase 6 della Borland, ma l'obiettivo finale è quello di utilizzare DB2 della IBM.
Fortunatamente il linguaggio di programmazione Java metta a disposizione per la connettività agli RDBMS i driver JDBC che rappresentano uno standard. Questo permette a parità di schema relazionale, vincoli di integrità, trigger, ecc. di realizzare delle applicazioni che siano il più possibile indipendenti dal RDBMS scelto e quindi è possibile passare da un database ad un altro effettuando nel caso più sfortunato ritocchi minimi al codice della applicazione.

Volendo schematizzare, i vincoli progettuali sono:

- Linguaggio di programmazione: Java 2
- Sistema operativo: Unix
- Web Server: Apache
- Application Server: Tomcat
- RDBMS: DB2

1.3 Specifiche Funzionali

Le informazioni seguenti sono state raccolte in particolare dalla attenta analisi del documento riguardante il progetto del sito web di facoltà . In questa tesi verrà analizzata e implementata ciò che in quel progetto è definito come pagina dei dipendenti, facente parte della sezione della didattica. Nella tabella riportata di seguito si possono vedere quali sono le informazioni necessarie per lo sviluppo completo della sezione della didattica nel portale web di facoltà.

Informazione	**Motivazione**
Attività collaterali quali Master, etc.	Master e Scuole di specializzazione pur conservando un carattere di "straordinarietà" fanno parte dell'offerta didattica della facoltà.
Descrizione dettagliata dell'offerta didattica	Queste informazioni coprono quasi interamente la categoria di informazioni relative alla didattica.
Esami ed Appelli	Gli esami fanno parte dell'attività didattica
Home page dei dipendenti	La Home Page di un professore deve poter contenere

	informazioni relative ai corsi tenuti, al materiale didattico e agli orari di ricevimento studenti.
Orario delle lezioni	
Spazio per ogni Insegnamento	

Tabella 1.1: Alcuni aspetti della didattica

Come si può vedere, il portale web di facoltà sarà dotato di numerose funzionalità. Affinché sia di facile gestione, molte della pagine del sito verranno realizzate in modo dinamico. Per ognuna di queste può essere necessario implementare anche alcune pagine di gestione delle informazioni da pubblicare. Ciò implica l'esigenza di realizzare più di una applicazione web per espletare ogni compito. Data la dimensione del progetto, è stata assegnata una priorità ad ogni applicazione da realizzare. Il valore di priorità è stata scelto sulla base della necessità di gestire le informazioni, e sull'impatto che questa applicazione ha sul sito. Il Web nasce come strumento per pubblicare documenti mentre le applicazioni web servono a organizzare e gestire informazioni. In questo progetto si tenta di introdurre nuovi strumenti per gestire le informazioni in modo efficiente e semplice, informazioni che ora sono gestite *manualmente*. Questo ovviamente ha un grosso impatto sull'organizzazione interna in quanto condiziona fortemente le abitudini di lavoro.
Un possibile sequenza temporale delle applicazioni web da introdurre nel sito potrebbe essere la seguente:

1. Home page dei dipendenti
2. Elenco delle pubblicazioni
3. Gestione spazio per ogni insegnamento
4. Notizie e documenti

La nuova applicazione permetterà di diffondere la conoscenza, la capacità di interazione con il Web server di Facoltà e permetterà di apprezzarne pregi e difetti.
Una volta familiarizzato con il sistema, il passaggio all'uso di informazioni già trattate sarà più semplice ed indolore. Sarà così possibile introdurre applicazioni quali la gestione delle *Informazioni sui Corsi*, l'*Elenco delle Pubblicazioni*, gli *Orari delle Lezioni*, ecc..
Per un certo periodo di tempo, fino all'introduzione dell'apposita applicazione Web, vi saranno informazioni prevalentemente statiche.
Occorrerà poi prevedere un responsabile che si preoccupi di pubblicare tali informazioni sul sito.

1.3.1 Home page dei dipendenti

Le home page dei dipendenti devono costituire un insieme di informazioni per inquadrare competenze, studi e ambito di ricerca per ogni docente, informazioni di utilità per gli studenti come ad esempio l'ubicazione dell'ufficio e l'orario di ricevimento. Inoltre supposto che un docente tenga uno o più insegnamenti è possibile per ognuno di questi, mantenere informazioni come avvisi, date degli appelli e materiale didattico. Inoltre sarebbe molto utile realizzare la funzione di iscrizione on-line agli esami.
Come si può notare le informazioni che è possibile pubblicare nella home page di un docente sono numerose, tra queste però è importante distinguere quali sono quelle istituzionali e quali quelle di utilità che il docente stesso può modificare.
Le informazioni istituzionali sono ad esempio nome, cognome, ruolo, e sono informazioni mantenute nella base di dati della Segreteria dell'Ateneo e sono modificabili solo da quest'ultima.
Per quanto riguarda gli insegnamenti, si possono considerare informazioni istituzionali il programma dell'insegnamento, le date degli appelli, ecc.
Le informazioni non istituzionali invece sono quelle che il docente può modificare liberamente, come ad esempio l'ambito di ricerca, eventuali affiliazioni e cooperazioni, per ogni insegnamento tenuto inoltre il docente può decidere di pubblicare tutto il materiale didattico che ritiene utile per la preparazione dell'esame e tutti gli avvisi necessari per consentire agli studenti di rimanere aggiornati sui risvolti dell'insegnamento. Come già detto non modificabile dal docente invece sono gli appelli degli insegnamenti che sono di competenza della segreteria, la quale deve naturalmente stabilire in base alla disponibilità e al numero di studenti iscritti l'aula dell'esame.

Le informazioni istituzionali e quelle di utilità devono quindi essere trattate in modo diverso, le prime con bassa frequenza di aggiornamento e con carattere di ufficialità non possono essere modificate dal dipendente ma solo dalla Segreteria, mentre le informazioni "utili" possono essere aggiornate più frequentemente e sono di responsabilità del singolo dipendente che dovrà poterle modificare facilmente. Per la gestione propria delle informazioni da pubblicare sulla home page dei dipendenti è necessaria una applicazione che permetta ai singoli "dipendenti" di modificare facilmente, via Web, le informazioni "utili", quindi in questo contesto il docente risulta essere autore, pubblisher e responsabile della informazione da lui stesso pubblicata. Si può prevedere che la frequenza di aggiornamento della pagina dei docenti sia settimanale, in quanto specie in

vicinanza di esami o durante il periodo delle lezioni si può fare della propria home page un vero e proprio punto di riferimento per un determinato insegnamento, consentendo anche agli studenti impossibilitati a partecipare ai corsi di rimanere aggiornati sullo stato d'avanzamento delle lezioni, pubblicando periodicamente avvisi e materiale didattico che facilitino la comprensione della materia. Naturalmente l'accesso a queste informazioni è consentito almeno in lettura a tutti i visitatori del sito web, per quanto riguarda l'iscrizione agli esami è possibile consentire solamente agli studenti di mettersi in lista a patto che questi vengano identificati in modo univoco ad esempio attraverso username e password, oppure ad esempio attraverso il numero di matricola.

Per la realizzazione della home page dei docenti, data la natura duplice (istituzionali e di utilità) delle informazioni da pubblicare è ovviamente necessario predisporre delle garanzie di accesso particolari ad un utente detto "amministratore" che sia in grado di variare anche i contenuti istituzionali. Questa figura coinciderà ovviamente con la Segreteria di facoltà.

La tabella mostra quali sono le informazioni istituzionali e quali quelle di utilità, nella pagina dei docenti.

Informazione	Istituzionale	Utilità
Nome e Cognome	X	
Ruolo	X	
Fax	X	
eMail	X	
Telefono	X	
Ubicazione Ufficio	X	
Orario di Ricevimento	X	
Orario delle lezioni	X	
Affiliazioni		X
Ambito di Ricerca		X
Collaborazioni		X
Incarichi		X

Cooperazioni		X

Tabella 1.2: Home Page Dipendenti: Informazioni istituzionali e di utilità

La gestione (inserimento, cancellazione e modifica) degli insegnamenti per ogni docente è lasciata interamente all'amministratore. Quindi tutti i dati sono di carattere istituzionale.

Informazione	Istituzionale	Utilità
Nome dell'insegnamento	X	
Anno Accademico	X	
Programma	X	
Descrizione	X	
Eventuali Propedeuticità	X	

Tabella 1.3: Informazioni istituzionali negli insegnamenti

Per ciò che riguarda gli appelli definibili per ogni insegnamento, si può considerare informazione di utilità solamente le eventuali note che il docente decide di aggiungere come completamento dell'informazione.

Informazione	Istituzionale	Utilità
Data Appello	X	
Aula	X	
Tipo	X	
Note		X

Tabella 1.4: Informazioni istituzionali negli appelli

Avvisi e materiale didattico per ogni insegnamento sono invece informazioni di utilità gestite direttamente dal docente. Le tabelle seguenti mostrano una possibile tipologia di informazioni pubblicabili tramite gli avvisi o il materiale didattico.

Informazione	Istituzionale	Utilità
Data Avviso		X
Titolo		X
Autore		X
Descrizione		X

Tabella 1.5: Informazioni di utilità negli avvisi

Gli avvisi in linea teorica possono essere utilizzati per pubblicare qualsiasi informazione, dai risultati di un appello allo stato di avanzamento del corso, oppure per rispondere a domande frequenti degli studenti.

Per ciò che riguarda il materiale didattico:

Informazione	Istituzionale	Utilità
Descrizione		X
Data		X
File		X

Tabella 1.6: Informazioni istituzionali nel materiale didattico

Anche in questo caso le funzioni offerte dalla applicazione che gestisce il materiale didattico pubblicato possono essere utilizzate da parte del docente, come meglio crede, ad esempio è possibile integrare le nozioni del corso con la pubblicazione di dispense, oppure con l'insieme dei testi d'esame degli appelli precedenti.

1.3.2 Gestione Spazio per ogni insegnamento

Una delle prime applicazione web che saranno introdotte del portale è proprio la gestione dello spazio sul server necessario per ospitare ad esempio il materiale didattico per ogni insegnamento.
E' di utilità fondamentale infatti quella di rendere disponibile agli studenti di un dato corso tutto il materiale didattico disponibile in formato elettronico, quali dispense ed esercizi limitando l'accesso ad esempio ai soli studenti. Si tratta quindi di realizzare aree del sito Web in cui i docenti possono depositare e descrivere "documenti" e dalle quali gli studenti possono scaricare tali documenti.
Il problema si riduce al riconoscimento (autenticazione) dei vari professori responsabili dei corsi, degli studenti abilitati, alla gestione di un'area (su un unità di memorizzazione di massa di un qualche server) in cui depositare e leggere i documenti e inserire adeguati commenti per la loro descrizione. I professori responsabili dei corsi potranno aggiungere, modificare o cancellare informazioni, gli studenti "autorizzati" potranno legge le informazioni depositate dai professori. Anche in questo caso se l'obiettivo è quello di realizzare una pagina web di facile consultazione e coerente, è necessario realizzare anche la relativa applicazione web per gestire i contenuti di queste pagine, infatti i metodi standard (ad esempio quelli basati su FTP) non soddisfano a pieno i requisiti necessari per la realizzazione di una applicazione completa, ma soprattutto funzionale.

1.3.3 Meccanismo di autentificazione utenti

Data la sensibilità delle informazioni contenute, nella home page docenti è necessario prevedere dei meccanismi di autentificazione degli utenti che siano il più possibile sicuri. Nasce dunque l'esigenza di identificare ogni utente che accede nel sito in modo univoco. Ogni membro della Facoltà, studente e collaboratore riconosciuto dalla Facoltà deve avere uno *username* ed una *password* tale da permettere un'autenticazione univoca e non *negabile* verso il sistema ogni qual volta accede a servizi "riservati" agli utenti della Facoltà. Deve esistere un particolare utente, ad esempio *anonimous* o *guest* assegnato di default alle connessioni (o meglio alle sessioni) non ancora autenticate. Questo particolare utente è utile per l'accesso alle informazioni pubbliche. Nel momento in cui si desidera accedere a particolari sevizi occorre autenticarsi in modo che il sistema possa verificare se l'utente autenticato ha il diritto di accedere al servizio richiesto. L'accesso a zone e servizi riservati è gestibile attraverso un meccanismo a *gruppi*. L'accesso ad una data area protetta del sito o a particolari funzionalità delle applicazioni della Facoltà sarà consentito solo a quegli utenti che appartengono ad un dato gruppo. Esempi di gruppi sono "*membro del consiglio di*

facoltà", "*preside*", "*segretario*","*docente*" .. Un utente può appartenere a più gruppi, e questo permette di ritagliare i diritti di accesso alle varie aree su misura dei singoli utenti. Naturalmente questa situazione è quella ideale, che si deve realizzare al completamento del sito web di facoltà. Al fine di ottenere un servizio di autentificazione degli utenti così dinamico e in grado di adattarsi ad ogni esigenza, è necessario realizzare una applicazione ad hoc per la gestione di gruppi di utenti, privilegi, username e password, ma questo esula dagli obiettivi di questa tesi.

Tuttavia la introduzione di un sistema di autentificazione di base per la realizzazione delle pagine dei dipendenti, anche se provvisorio, deve essere realizzato, altrimenti non è possibile gestire le informazioni memorizzate nel sito in modo sicuro. In questo particolare contesto risulta sufficiente discriminare gli utenti che appartengono al gruppo dei docenti e quelli che appartengono al gruppo della segreteria, che hanno funzione di amministratori per quanto riguarda le informazioni di carattere istituzionale. Inoltre si deve dare la possibilità agli amministratori di inserire ed eliminare account per i docenti in modo da compensare una eventuale variazione futura del personale dell'Università.

Capitolo 2
Analisi dei requisiti

2.1 Metodologie di Analisi

La necessità di realizzare applicazioni che siano facilmente mantenibili, che siano realizzate con coerenza e ben concepite, implica che venga sempre eseguita una accurata fase di analisi del progetto. Ovviamente essendo questa una fase descritta in praticamente tutte le metodologie che trattano del ciclo di vita di una applicazione, è nata l'esigenza di realizzare degli strumenti e delle tecniche di analisi efficienti e di facile utilizzo. La scelta della tecnica da utilizzare per effettuare l'analisi del progetto deve essere fatta in base a numerosi fattori, che dipendono dalle specifiche funzionali, dal linguaggio di programmazione dal tipo di progetto ecc. Naturalmente utilizzare un tipo di analisi piuttosto che un altro può portare in un preciso contesto a numerosi vantaggi.

2.1.1 Object Oriented Analysis And Design

Esistono differenti strumenti di analisi, da quelli informali a quelli formali. Ad esempio:

?? Informali
?? Semiformali
?? Formali

Gli strumenti informali sono quelli per cui la sintassi e la semantica non sono definite in modo rigoroso. Un possibile strumento informale e costituito dalle checklist. In genere può essere utilizzato quando la conoscenza del problema da parte dell'analista è più che ottima.
Infatti più lo strumento di analisi è approssimativo e maggiore è la possibilità di una interpretazione personale del problema, con conseguente possibilità di errore.
Gli strumenti semiformali si pongono a metà tra le due categorie, sono largamente usati, molto efficienti e permettono di realizzare in breve tempo schemi grafici con un alto contenuto informativo. Un tipico esempio di analisi semiformale è data per quanto riguarda gli schemi dei database dallo schema Entity/Relationship.

Infine l'analisi formale e quella realizzata con tecniche la cui semantica e sintassi sono definite in modo rigoroso spesso quasi come si trattasse di un linguaggio di programmazione.
In generale nello sviluppo di applicazioni complesse o di una certa importanza non vengono mai utilizzati tecniche di analisi informali, ma solo tecniche di analisi formali e semiformali. Inoltre la fase di analisi produce documentazione e quanto più questa è accurata tanto più utile risulterà in caso di problemi o di revisioni del software.
La diffusione dei linguaggi di programmazione Object Oriented, ha fatto si che anche gli strumenti di analisi si spostassero in tale direzione, infatti adesso le tecniche di analisi più recenti sono quelle fornite da metodologie Object Oriented Analysis (OOA). Utilizzare uno strumento di analisi OO e un linguaggio di programmazione OO è senza dubbio un connubio vincente e in questi casi si parla di Object Oriented Analysis & Design (OOAD) . I migliori strumenti di analisi sono in grado di realizzare, ad esempio, i diagrammi delle classi della applicazione. Inoltre forniscono i tipici benefici di una metodologia orientata agli oggetti:

- ?? Mantenibilità: è ottenuta grazie alla corrispondenza degli oggetti con il mondo reale facilitando l'individuazione delle astrazioni logiche
- ?? Riusabilità: quando è possibile si è in grado di sfruttare analisi ed esperienze acquisite in precedenza in un progetto con uno scenario implementativo differente

Le metodologie di Object Oriented Design consentono, a partire dai risultati forniti dall'analisi OO di ottenere un Design Model da utilizzare come riferimento durante la fase implementativi, consentendo in un certo senso la prosecuzione della fase di analisi. Anche in questo caso gli obiettivi del Object Oriented Design sono la facilità di manutenzione e il riuso del codice prodotto. Una buona strategia aziendale per lo sviluppo di applicazione infatti, è quello di mantenere un repository di tutte le classi e i diagrammi (sia dal punto di vista OOA che OOD) prodotti al fine di facilitare lo sviluppo di applicazioni future.
Concludendo si può affermare che la fase di analisi è molto importante perché fornisce una rappresentazione schematica delle specifiche dei requisiti, produce cioè della documentazione che è possibile visionare durante le fasi successive del progetto, e se si utilizza un linguaggio di modellazione OO, è possibile utilizzare anche in progetti successivi.

2.2 Introduzione all' UML

2.2.1 Cenni Storici

La tecnica di analisi Unified Model Language (UML) è relativamente recente ed è stata realizzata dall'unione delle precedenti tecniche degli ormai famosissimi Booch, Rumbaugh e Jacobson. UML risulta quindi una creazione collettiva, nata con l'esigenza di fornire uno strumento unico per l'analisi, efficiente, e che copra ogni possibile aspetto di un sistema (teoricamente anche non software). Alla fine degli anni 80 esistevano diverse decine di metodologie di tecniche di OO, ma Booch, sotto la spinta della Rational in cui lavorava, venne incaricato della realizzazione di un progetto che unificasse le più efficienti tecniche esistenti. Da li a poco anche Jim Rumbaugh venne assunto da Rational portando alla società tutte le conoscenze relative alla diffusa tecnica OMT (Object Modelling Tecnique). Anche l'Object Management Group (l'associazione che si occupa di definire gli standard per ciò che riguarda le tecniche ad oggetti) spingeva molto affinché venisse prodotto un solido metodo OO, e fornirono agli autori di UML uno schema preciso di ciò che volevano ottenere, in modo tale da definire una semantica precisa. Nel gennaio del 1997 la Rational rilasciò la versione 1.0 della documentazione di UML come contributo all' OMG. Fondamentale in questo panorama fu anche il coinvolgimento di grandi aziende del campo informatico come IBM, Microsoft, HP, Oracle e altre, tutte molto interessate a questo nuovo tipo linguaggio di analisi. UML nasce dunque anche per una esigenza commerciale, non solo tecnica. In figura 2.1 è possibile vedere l'andamento temporale dello sviluppo del linguaggio UML

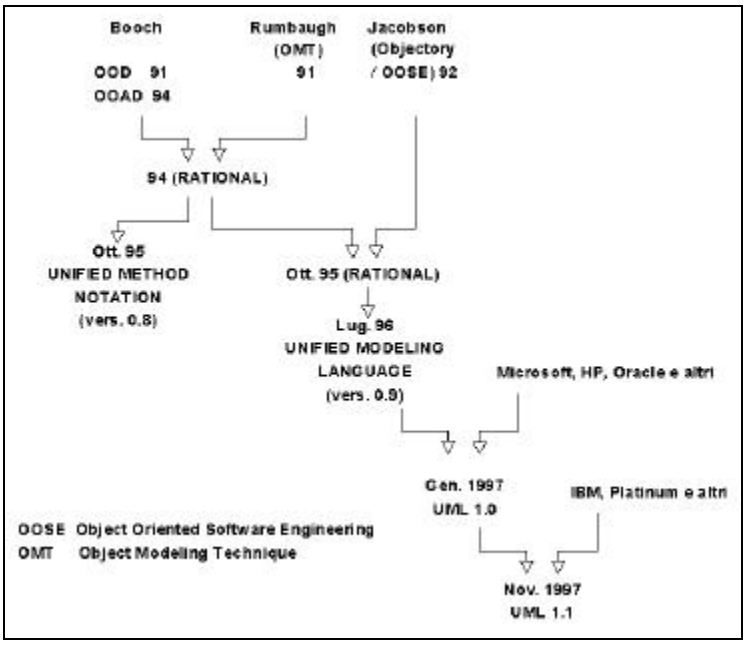


Figura 2.1: Evoluzione di UML

2.2.2 Diagrammi Significativi

Di seguito si cercherà di fornire alcuni concetti di base su UML, non si ha la pretesa di essere esaustivi in quanto per descrivere interamente questo linguaggio di modellazione forse non basterebbe un libro. L'obiettivo che ci si pone è quello di inquadrare i diagrammi principali e le loro modalità di utilizzo nello sviluppo di applicazioni. Come già detto UML è un linguaggio di progettazione non è un modello, e può rappresentare qualsiasi sistema software, inoltre da novembre 1997 rappresenta uno standard definito dal OMG. UML è un linguaggio di progettazione universale in grado di rappresentare sistemi molto diversi senza differenze legate alla tecnologia: dai sistemi web a quelli più tradizionali o alle applicazioni più datate, come ad esempio, quelle realizzate in linguaggi strutturati (non orientati agli oggetti), come il Cobol o il Pascal.
La maggior informazione rappresentata da UML è attraverso una notazione standard basata su un meta-modello integrato delle componenti che costituiscono il sistema software. Questo linguaggio è costituito da diversi diagrammi, ma non viene prescritta una sequenza di realizzazione, cioè non c'è una priorità tra le attività poste alla realizzazione dei diagrammi e ciò permette a più gruppi di lavoro di realizzare l'analisi. Come è già stato accennato UML non è nient'altro che la naturale evoluzione di modelli precedenti, per tale motivo ha forti analogie con altri schemi molto noti (ad esempio E/R, OMT, ecc). UML è basato su un meta-modello integrato, composto da numerosi elementi collegati tra loro, secondo regole precise, la maggioranza di questi elementi ha una rappresentazione grafica. Infatti l'aspetto grafico costituisce un punto di forza del modello perché consente immediatamente di avere una visione globale di ogni parte costituente il sistema, e dato che i diagrammi sono molto intuitivi consente di fare capire l'applicazione anche a chi non conosce a fondo il modello.

I diagrammi di UML sono:

- ?? Use Case
- ?? Class Diagram
- ?? Object Diagram
- ?? Sequence Diagram
- ?? Collaboration Diagram
- ?? State Diagram
- ?? Activity Diagram

?? Component Diagram
?? Deployment Diagram

Lo standard non definisce ne la sequenza di utilizzo dei diagrammi, ne il numero di quelli da utilizzare, la scelta viene lasciata agli analisti del sistemi, al grado di dettaglio che si vuole ottenere e alla reale necessità di rappresentazione. Ovviamente se il numero di diagrammi che si scegli di utilizzare è molto elevato la precisione dell'analisi aumenta, ma il beneficio introdotto può comunque non essere significativo per quel particolare contesto.

Di seguito verranno trattati alcuni dei diagrammi più significativi nello standard UML, gli use case, il class diagram, il sequenze diagram.

Use Case

Lo scopo dell'analisi è di trasformare i requisiti utente in un insieme di Use Case che descrivono le tipiche interazioni con il sistema in costruzione, e in un modello ad oggetti ad alto livello del dominio del problema. L'analisi dovrebbe focalizzarsi esclusivamente sul problema e dovrebbe essere fatta senza prendere in considerazione la successiva implementazione.
Gli Use Case non dovrebbero considerare il "sistema" come un unico blocco; viceversa dovrebbero mettere in evidenza le interazioni fra i diversi "ruoli" degli attori reali, cioè delle persone coinvolte nella realizzazione del processo nel mondo fisico. Questo permette, nella successiva fase di progettazione, di individuare corrispondenti "attori software" (oggetti), specializzati nelle attività individuate (controlli di validazione, scelte elaborative costituenti la logica dell'applicativo e dipendenti dalla politica aziendale che il programma rispecchia).

Uno tra gli schemi maggiormente realizzati dunque, è costituito dagli Use Case, che descrivono il comportamento del sistema o parte di esso tramite un insieme di azioni eseguite da uno o più attori . Un attore rappresenta un insieme coerente di *ruoli* che gli utenti giocano quando interagiscono con il sistema. Studiare i casi d'uso significa studiare le funzionalità che il sistema mette a disposizione dei suoi utilizzatori, pertanto è in genere il primo diagramma che viene realizzato in quanto aiuta a scoprire i requisiti.
Questo schema serve per catturare il comportamento futuro del sistema che si sta modellando **senza** specificare come verrà modellato, infatti viene specificato solo cosa fa il sistema nascondendo cioè i dettagli implementativi. La definizione del modello si ottiene identificandone gli elementi costituenti, attori, use case e relazioni. Le relazioni possono essere definite tra due use case, tra attori (generalizzazione), oppure tra un attore ed un use case.

Nel caso di applicazioni web gli use case ricalcano in parte gli schemi di navigazione del sito web. Se tuttavia supponiamo che il sito sia costituito da un certo numero di aree protette è bene mettere in evidenza negli use case quali sono gli attori che hanno accesso a tali aree ed eventualmente costituire una gerarchia tra gli attori per sottolineare questo aspetto. Gli use case possono anche essere utilizzati alla fine dello sviluppo della applicazione per verificare se questa esegue realmente i compiti che erano stati previsti in analisi, questo è molto importante perché può aiutare a scoprire malfunzionamenti oppure mancate funzionalità.

Di seguito viene proposta la notazione attuale degli use case nella versione dell'UML 1.1 (tabella 2.1)

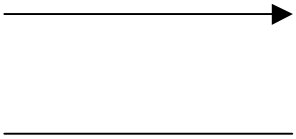
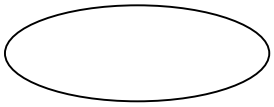
Simbolo	**Tipo**	**Descrizione**
	Attore	E' un utilizzatore del sistema, può essere un essere umano oppure un altro sistema.
	Relazione	E' una associazione tra attore e use case, oppure due o più tra use case oppure tra più attori. Puo' essere monodirezionale o bidirezionale, in quest'ultimo caso in genere non si disegnano le frecce.
	Use Case	E' una particolare modalità di utilizzo del sistema.

Tabella 2.1: Simbologia degli Use Case

In figura 2.2 potete vedere un semplice caso d'uso che mostra la relazione che intercorre tra un ipotetico cliente ed un venditore:

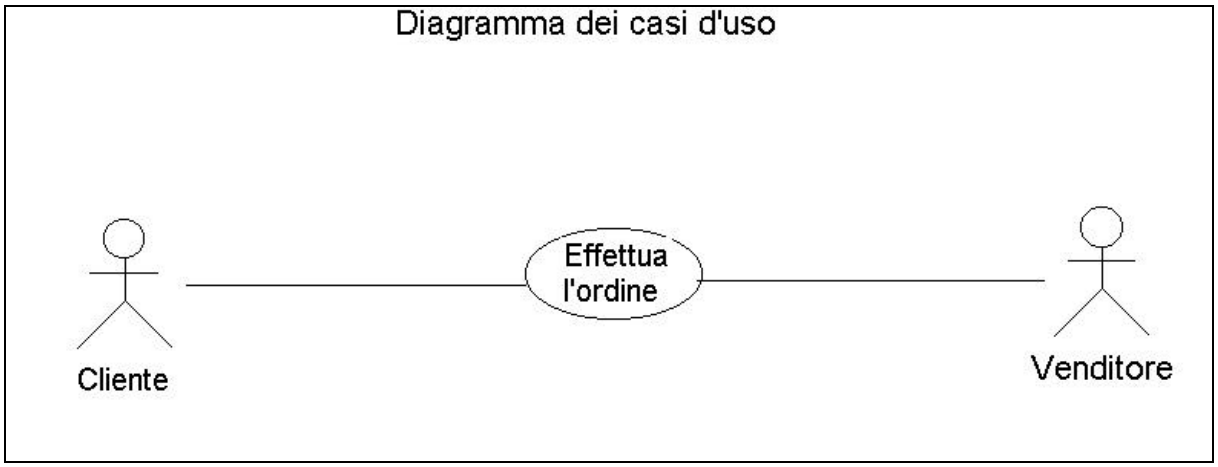


Figura 2.2: Esempio di Use Case

CLASS DIAGRAM

I diagrammi delle classi descrivono il tipo degli oggetti che compongono il sistema e le relazioni esistenti tra di loro. Le relazioni possono essere di due tipi:

?? Associazioni

?? Generalizzazioni

I diagrammi delle classi mostrano anche gli attributi e le operazioni (metodi) di una classe, le relazioni mostrano il modo con cui sono collegati gli oggetti. Invece la generalizzazione implementa il concetto noto con il nome di subclassing. La notazione utilizzata nel diagramma delle classi di UML deriva principalmente dal modello OMT di Rumbaugh. Una classe è rappresentata da un rettangolo diviso in tre parti, il nome della classe compare nella parte alta, gli attributi della classe nella parte centrale e i metodi nella parte in basso. Le relazioni sono rappresentate con una linea e la generalizzazione con un triangolo. Il diagramma delle classi fornisce una vista della applicazione che può essere:

?? **Concettuale**

?? **Specifica**

?? **Implementativa**

Spesso nella fase di analisi di prima istanza è inutile addentrarsi nei dettagli per cui la sola vista interessante risulta essere quella concettuale, che come dice la parola stessa specifica il funzionamento della applicazione senza esplicitare tutte le proprietà o tutti i metodi di una classe.

In figura 2.3 potete vedere un generico esempio di un diagramma della classi che rappresenta le relazioni che intercorrono tra un libro, l'editore e l'autore in una biblioteca che effettua dei prestiti. Si può notare inoltre la specializzazione del libro in "libro prezioso".

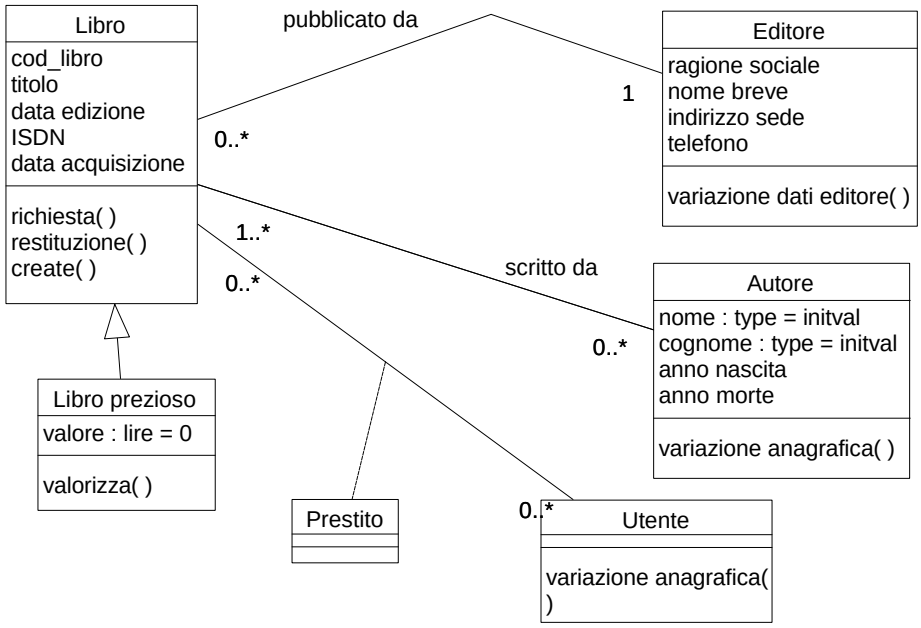


Figura 2.3: Esempio di Class Diagram

In una realtà implementativa, quando si ha a che fare con applicazioni che fanno uso di basi di dati, capita sovente che il diagramma della classi ricalchi in parte lo schema E/R e di conseguenza anche la realizzazione del codice mantiene una certa corrispondenza con questo schema, naturalmente questo è un vantaggio considerevole, in quanto permette facilmente di scoprire eventuali errori di programmazione.

SEQUENCE DIAGRAM

I diagrammi di sequenza insieme ai diagrammi di collaborazione fanno parte di UML sotto il nome di **diagrammi di interazione** (interaction diagram). Mentre gli schemi definiti in precedenza non forniscono informazioni temporali sulla sequenza delle azioni, ma forniscono solamente una visione

statica del sistema, i diagrammi di interazione descrivono lo svolgimento di una attività. Anche in questo caso lo standard UML non definisce in modo rigoroso quale dei due diagrammi utilizzare, ma lascia la possibilità di decidere all'analista del sistema. La scelta deve essere fatta in base a diverse considerazioni, prima fra tutte il grado di informazioni che un diagramma o l'altro riescono a fornire in quello specifico contesto. Eventualmente è possibile usare entrambi i diagrammi.
Il disegno degli Interaction Diagram permette di tracciare la "*collaborazione*" tra gli oggetti: un oggetto collabora con un altro se deve invocare uno o più metodi di quest'ultimo per assolvere ad una propria responsabilità. Quindi per ogni responsabilità di un oggetto, occorre determinare se questo è in grado di assolverla da solo o se viceversa deve utilizzare qualche altro oggetto, con cui quindi "collabora". I diagrammi di interazione dunque, mostrano come gli oggetti collaborano per realizzare una azione. Spesso viene fornita anche una descrizione della azione espressa in linguaggio naturale o in pseudocodice a fianco del diagramma.

La forza del diagramma di sequenza è proprio nell'essere immediatamente comprensibile anche senza conoscere il formalismo grafico che ne sta alla base. Tuttavia di seguito se ne fornirà una breve descrizione.
Ogni oggetto presente nel diagramma di sequenza viene disegnato attraverso una box posta nella parte alta dello schema, le interazioni tra gli oggetti, cioè i messaggi che gli oggetti si scambiano, sono rappresentati dalle frecce. Ogni oggetto è collegato ad una linea verticale, detta linea di vita, che rappresenta appunto la durata temporale dell'oggetto. I messaggi che gli oggetti si scambiano devono trovare riscontro nel diagramma delle classi, sono cioè i metodi degli oggetti. Spesso oltre al diagramma viene anche fornita una semplice descrizione testuale di ciò che scaturisce gli eventi nel diagramma che aiuta a chiarire il funzionamento della applicazione.

In figura 2.4 viene mostrato un generico diagramma di sequenza.

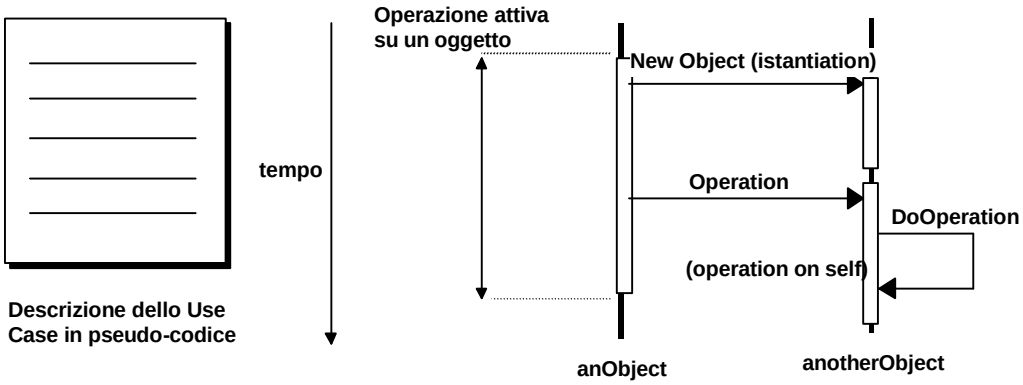


Figura 2.4: Esempio di Sequenze Diagram

2.3 Analisi della applicazione Web con UML

Volendo fornire una adeguata documentazione relativa alla applicazione web per il sito della facoltà di ingegneria informatica, la scelta è ovviamente caduta su UML, in quanto rappresenta lo standard attuale.

Come è stato specificato anche precedentemente, l'analisi attraverso il linguaggio di modellazione UML tocca molteplici aspetti di una applicazione fino a giungere ad un livello di dettaglio veramente notevole. In questa analisi verranno considerati solamente alcuni dei diagrammi disponibili per la modellazione con UML. La sequenza di realizzazione dei diagramma sarà la seguente:

- ?? Use Case
- ?? Class Diagram
- ?? Sequence Diagram

Ovviamente non è possibile effettuare una analisi completa e di dettaglio di tutto il progetto del sito web di facoltà nel tempo necessario per realizzare la tesi. Per tale motivo alcuni aspetti verranno trattati superficialmente, mentre per alcune parti del sito verrà realizzata un analisi di dettaglio, al fine di mettere in evidenza tutti i particolari che possono fornire un maggiore contenuto informativo e una maggiore comprensione di quello che sarà l'applicazione web. E' inoltre importante mettere in evidenza quale è il sistema da modellare, volendo evitare di ripetere ciò che è stato fatto in modo molto accurato nelle altre tesi (in particolare nella tesi svolta da Marzia Da Como), il punto di vista viene posto dagli utenti verso l'applicazione. Non si prende dunque come punto di vista l'intero sistema istituzionale (che peraltro sarebbe molto complesso) verso l'applicazione web, ma si mette solamente in evidenza come le varie tipologie di utenti interagiscono con l'applicazione realizzata.
Questo al fine di fornire con questa analisi anche una documentazione che possa in qualche modo servire anche come manualistica. Di seguito si vogliono descrivere le possibilità offerte dalla applicazione, perché su queste verrà maggiormente concentrata l'analisi, si fornisce dunque un riassunto molto sintetico e in linguaggio naturale delle specifiche dei requisiti.

In primo luogo è necessario consentire l'accesso all'area di gestione del sito web solo agli utenti autorizzati. Una volta discriminato l'accesso di un utente è necessario verificare a quale categoria appartiene, se è un docente o comunque fa parte del personale docente, si offre la possibilità di modificare i propri dati personali e di modificare i dati dei propri insegnamenti. Inoltre per ogni

insegnamento viene consentita la possibilità di gestire la lista degli appelli, la lista degli avvisi e del materiale didattico. Ogni cambiamento effettuato su questi dati modificherà in modo immediato la relativa home page. Ad un utente particolare (l'amministratore) viene inoltre data la possibilità di modificare alcuni dati istituzionali, di aggiungere e togliere account (e conseguentemente di eliminare le home page), di gestire username e password per gli atri utenti.
Il sito web è accessibile da chiunque in via di consultazione, in particolare se consideriamo gli studenti, viene fornita la possibilità di iscriversi agli esami on-line, e di consultare il materiale didattico e ogni altra informazione presente nella pagina del personale docente. Deve essere presente inoltre una pagina che fornisca la lista di tutti i docenti, consenta di contattarli via mail e di accedere alla loro pagina personale.

2.3.1 USE CASE

E' bene specificare alcune convenzioni utilizzate nel realizzare questo diagramma, in quanto alcune volte lo standard può non essere sufficientemente chiaro. Al fine di fornire un maggiore contenuto informativo, negli schemi si intende con una freccia bidirezionale o una linea la possibilità di per un attore di intergire con il caso d'uso del sistema sia in visualizzazione che in modifica delle informazioni. Altrimenti con una freccia singola con verso orientato al caso d'uso, la possibilità di leggere solamente il contenuto della pagina, senza la possibilità di effettuare modifiche. Ad esempio con riferimento ad una ipotetica home page del sito web di facoltà, il relativo use case si può schematizzare nel modo seguente (figura 2.5):

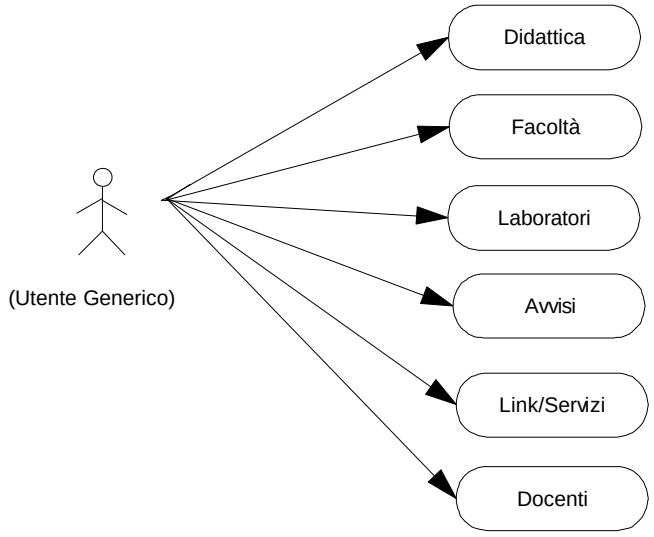


Figura 2.5: Use Case Home Page del sito

Per chiarire lo schema in figura è bene fare qualche considerazione. Allo stato attuale di sviluppo la home page del sito web di facoltà è statica, ciò implica che il suo contenuto non può essere modificato dalla applicazione web, per tale motivo le frecce vanno dall'attore verso gli use case specificando che l'interazione avviene in modalità di sola lettura. Inoltre essendo una home page di un sito di pubblica utilità è visualizzabile da qualsiasi persona, non sono cioè previste restrizioni di accesso e pertanto non risulta necessario specificare a quale tipologia di utenti appartiene l'attore, ma lo si indica come "utente generico". Se fosse stata invece una pagina il cui accesso era consentito solamente da una specifica categoria di utenti si sarebbe reso necessario specificare quale era questa categoria.
Anche se non è messo in evidenza dal relativo use case, alcune pagine del diagramma precedente possono essere realizzate in modo dinamico, ad esempio la pagina *Docenti* è dinamica e mostra una lista di tutto il personale e le relative e-mail, consentendo di raggiungere la home page del docente a partire dal nome di questo. Come già detto in precedenza gli use case nascondono i dettagli implementativi.

Di seguito si modella la parte di accesso al sito web, cioè l'area di Login (figura 2.6). Tale form è ovviamente accessibile da un URL ben precisa, che per ovvi motivi di sicurezza non viene indicata in questo documento. Attualmente sono previsti due livelli di accesso, l'accesso dell'amministratore e l'accesso del docente, a seconda dell'username e password inseriti viene discriminata la tipologia di utente.

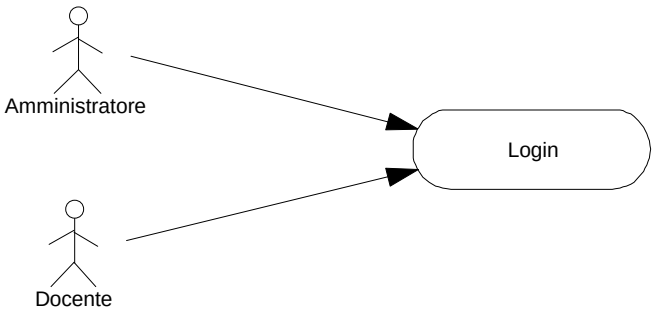


Figura 2.6: Use Case Area di Accesso

L'amministratore come già detto nella specifica dei requisiti, ha la possibilità di aggiungere account ai docenti, ad ogni account sostanzialmente corrisponde una pagina web per il docente. Inoltre ha la possibilità di inserire ed eliminare gli insegnamenti per ogni docente, gestire le password, eliminare un account. L'amministratore può inoltre modificare le informazioni di carattere istituzionale presenti nella pagina dei docenti secondo quanto è stato definito nella specifica dei requisiti. E' importante sottolineare che, in questo contesto, l'amministratore ricopre il solo ruolo di **pubblisher** dell'informazioni, in questo caso cioè deve essere solo l'esecutore delle operazioni di inserimento e cancellazione, tali operazioni ovviamente non devono essere fatte di sue spontanea iniziativa, ma devono essere autorizzate dalle opportune autorità.

Per evitare di rendere eccessivamente complicati gli use case si indica con *gestione* la possibilità di effettuare operazioni di lettura,inserimento, cancellazione e modifica di informazioni. Laddove queste operazione non sono completamente consentite si userà il solo termine cancellazione o modifica o inserimento. Sostanzialmente l'use case gestione rappresenta l'insieme di operazioni mostrate in figura 2.7.

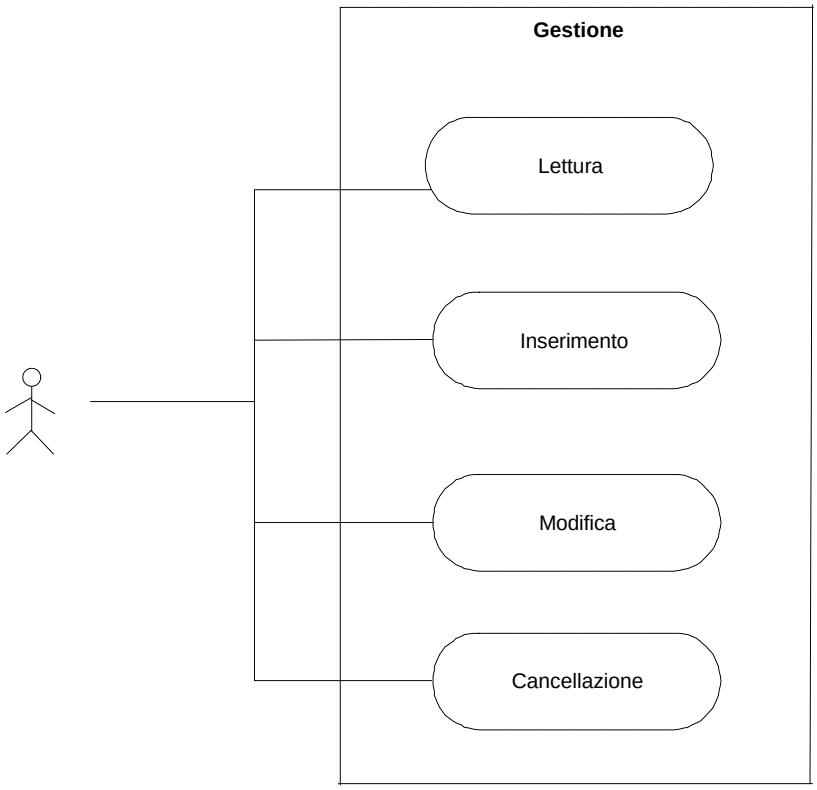


Figura 2.7: Rappresentazione Operazioni

Tornando alle possibilità offerte all’amministratore, il corretto use case è rappresentato il figura 2.8

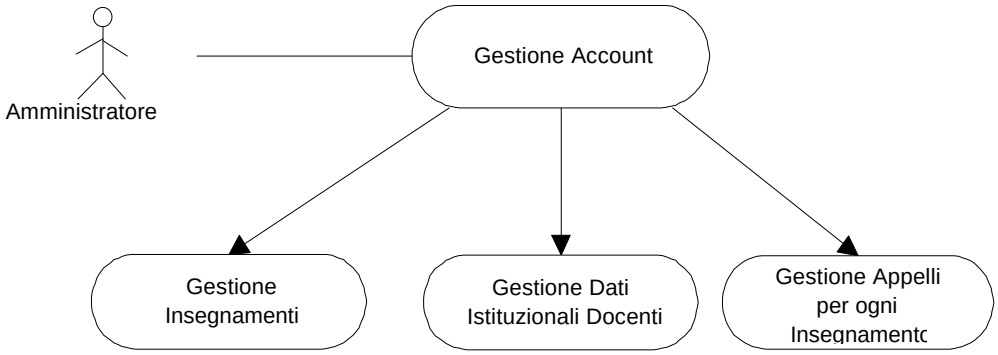


Figura 2.8: Operazioni consentite all'amministratore

Più complessa è invece la situazione relativa alla gestione delle informazioni per un docente. Quest'ultimo può infatti modificare i propri dati personali con carattere di utilità in qualsiasi momento, modificare i dati relativi ai propri insegnamenti, e per questi gestire materiale didattico, e avvisi. In questa situazione il docente risulta ricoprire il ruolo di autore dell'informazione e anche di pubblisher. Inoltre ogni modifica effettuata da docente avrà influenza immediata sulla proprio Home Page quando questa viene consultata, nello use case bisogna dunque mettere in evidenza la possibilità da parte di un utente generico che visita il sito della facoltà di visualizzare la home page del docente.

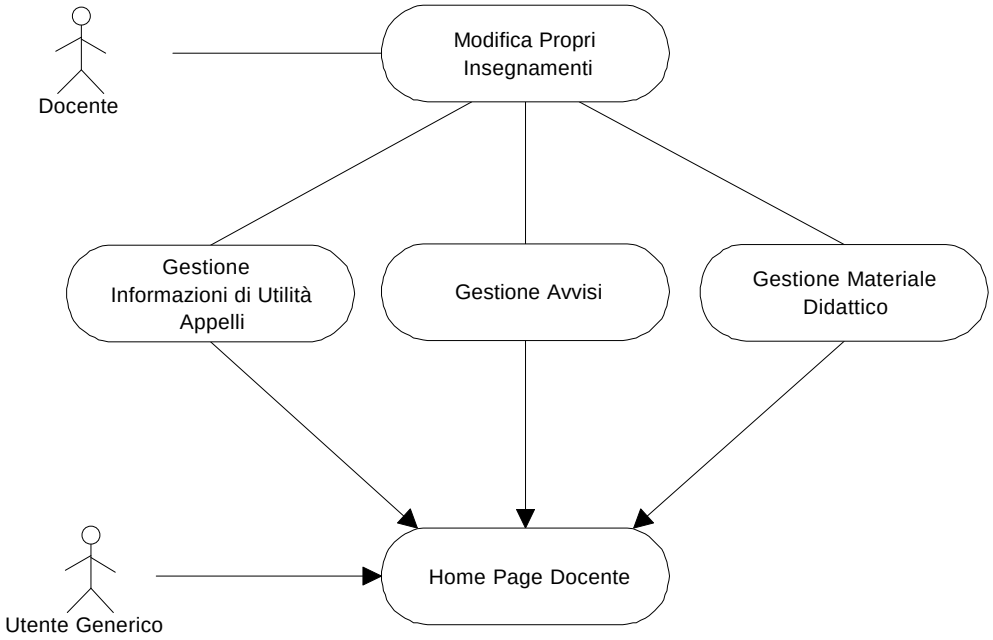


Figura 2.9: Operazioni consentite al docente

Cambiamo per un attimo il punto di vista e consideriamo l'attore studente, questo dovrebbe avere la possibilità di effettuare l'iscrizione On-Line agli esami, tale operazione viene effettuata tramite lo username e la password dello studente o qualsiasi altro metodo equivalente che lo identifichi in modo univoco (una possibilità alternativa può essere costituita dall'accoppiata matricola e nome dello studente). L'iscrizione On-Line agli esami avviene dalla pagina del docente ed è rappresentata dallo use case in figura 2.9. In questo caso il ruolo che lo studente ricopre è quello dell'autore e pubblisher dell'informazione, in quanto è esso stesso che compie l'atto di materiale di iscriversi e ne esprime la volontà, inoltre risulta essere anche responsabile dell'informazione pubblicata.

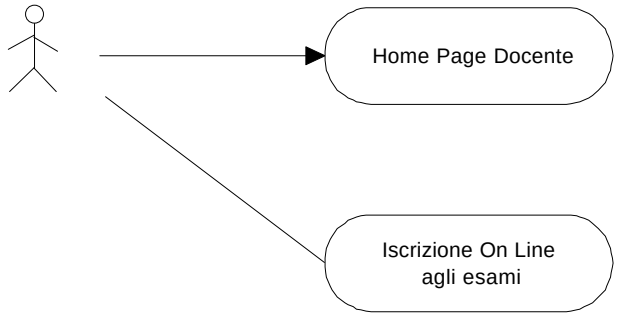


Figura 2.10: Use Case Iscrizione On-Line agli esami

Con questo schema si è conclusa la parte di analisi con gli use case relativa alla applicazione realizzata per il sito della facoltà di Ingegneria Informatica. E' evidente che essendo presa in considerazione solamente una porzione limitata della realtà da modellare, alcuni concetti possono essere soggetti a leggere modifiche. Inoltre il progetto del sito web di facoltà è molto vasto e prevede passi successivi per essere completato, tra questi però la realizzazione della home page docenti era il primo, e quindi per non violare i vincoli progettuali si è deciso di iniziare la realizzazione implementativi proprio da questo.

Capita sovente che durante l'analisi con gli use case, nel caso vengano coinvolti più attori, sia possibile realizzare uno schema che mette in evidenza le gerarchie tra questi. Lo schema proposto in figura 2.10 è parziale e dedotto solamente in base alla analisi fatta svolta fino a questo momento.

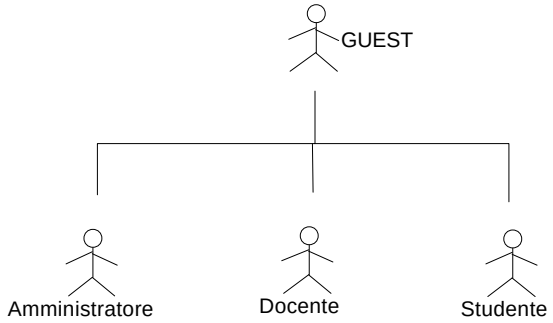


Figura 2.11: Use Case di prima generalizzazione

Uno schema come quello di figura 2.10 prende il nome di generalizzazione, l'attore "Guest" vuole rappresentare il visitatore del sito web di facoltà, è l'entità meno specializzata, sostanzialmente non può fare altro che leggere le informazioni pubblicate sul sito, senza alcuna possibilità di modifica.
E' innegabile che la situazione reale sia ben più complicata di quella proposta, ma per coerenza con il progetto era doveroso mettere in evidenza anche questo aspetto.

Di seguito, anche se attualmente non sono stati implementati, si vogliono mettere in evidenza gli use case relativi ad importanti aspetti del sito web di facoltà: gli avvisi e la didattica. Gli avvisi (cioè le News) devono potere essere visualizzati da chiunque acceda al sito, ma devono essere gestiti solamente da personale specifico. In questo use case non si mettono in evidenza i ruoli perché si vuole dare solamente una visione di ciò che sarà l'applicazione una volta completata, per ulteriori informazioni sui ruoli degli attori presenti si rimanda alla tesi [1].

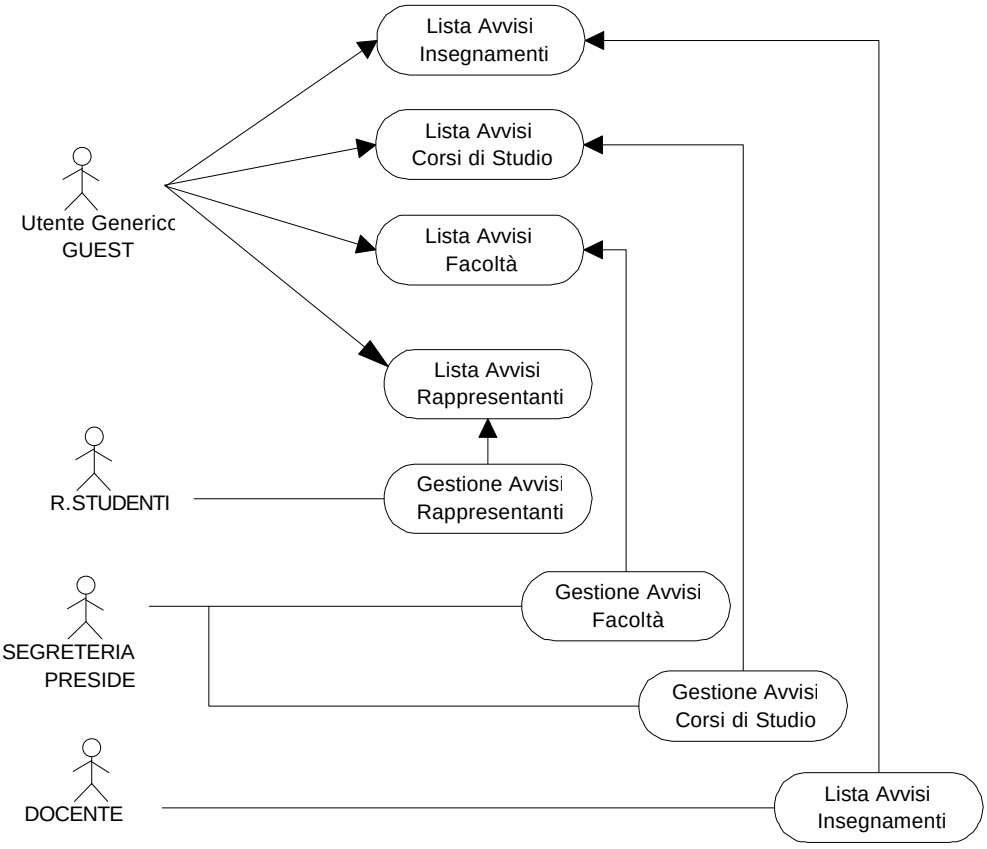


Figura 2.12: Use Case di gestione avvisi

Come si può notare dalla figura 2.11 la situazione reale per quanto riguarda le tipologie di attori è più complicata rispetto a ciò che aveva messo in evidenza la sola porzione di analisi relativa alla

applicazione per la realizzazione delle pagine web dei docenti. In questo caso è conveniente esprime una generalizzazione degli att ori come segue (figura 2.12):

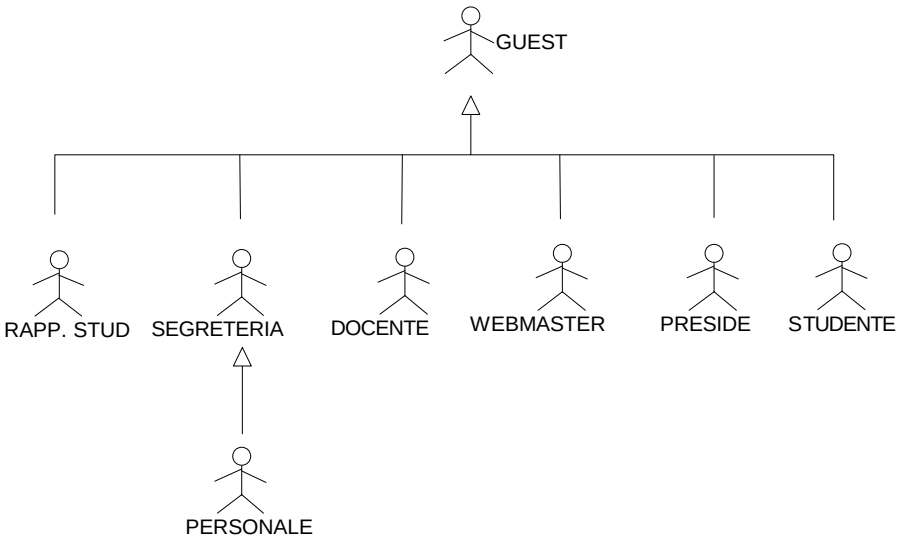


Figura 2.13: Generalizzazione degli attori

Lo use case dopo l'introduzione della generalizzazione diventa il seguente (figura 2.13):

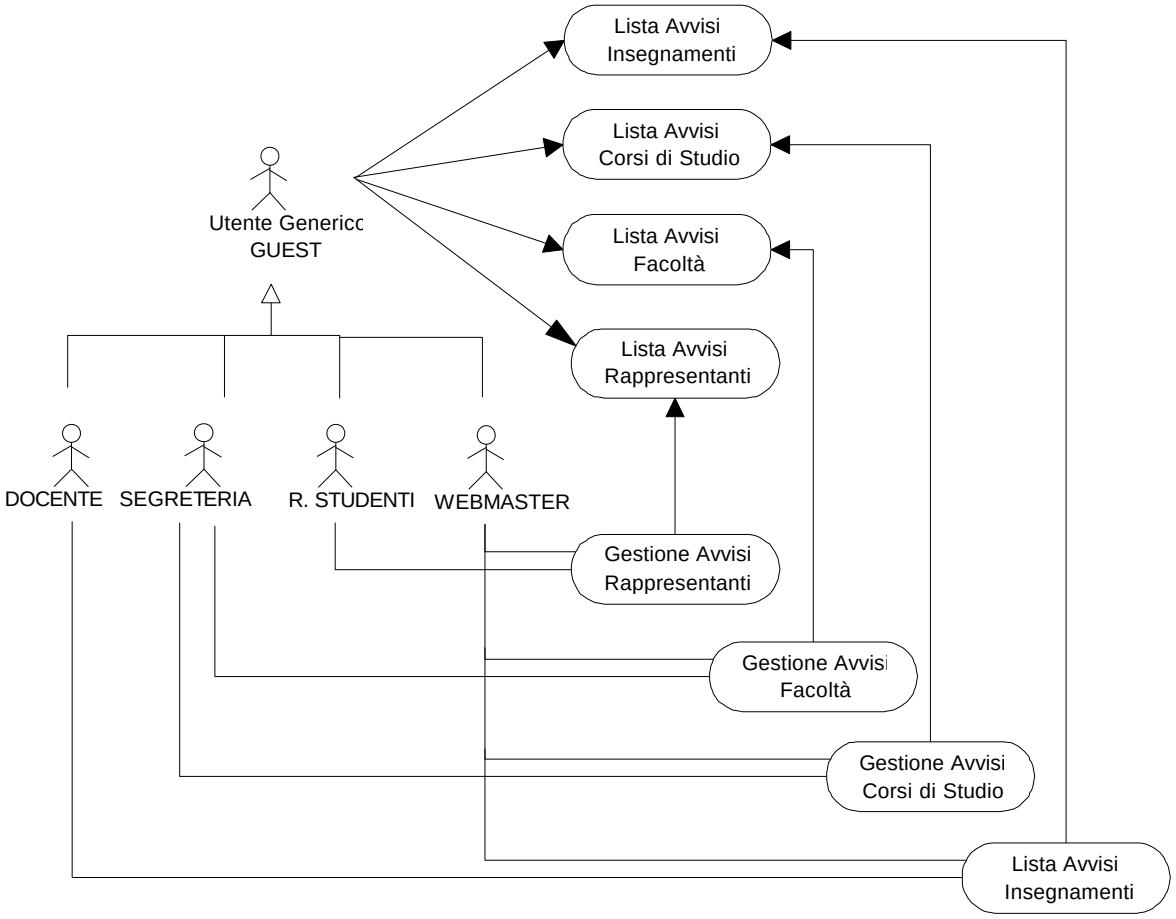


Figura 2.14: Use Case completo per la gestione avvisi

Un possibile schema esaustivo che mette in evidenza gli aspetti relativi alla didattica è quello di figura 2.14.

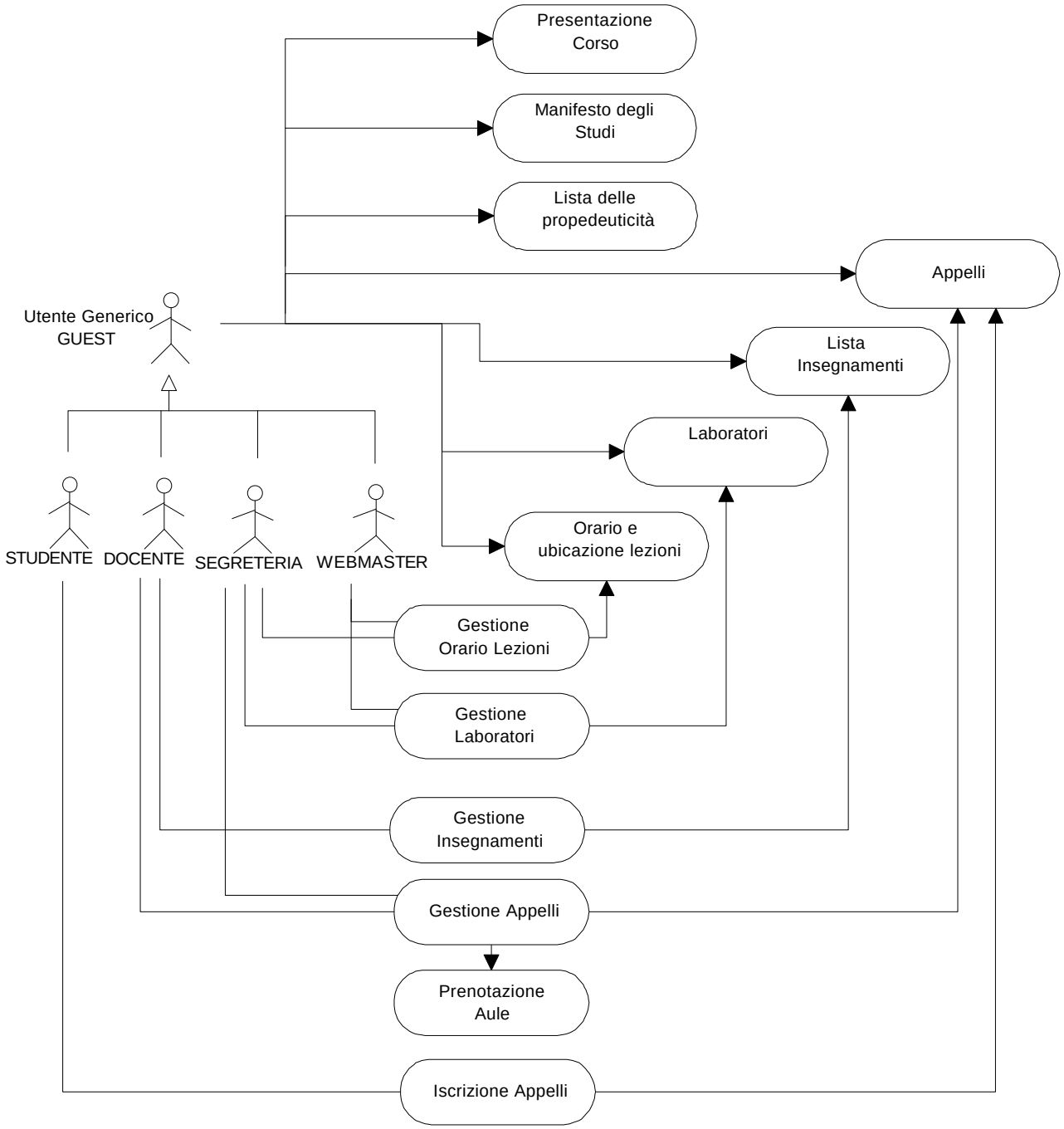


Figura 2.15: Use Case per la gestione della didattica

2.3.2 CLASS DIAGRAM

Molte informazioni relativi al diagramma delle classi si possono ricavare dal progetto del sito web di facoltà, che presenta questo aspetto in modo molto esaustivo. Il livello di dettaglio a cui si vuole arrivare è quello di massima, in quanto in questo documento non avrebbe praticamente significato descrivere ogni singolo metodo o proprietà di tutte le classi presenti, si vuole dare piuttosto una visione di insieme di ciò che sono le classi in gioco e le relazioni che intercorrono tra di esse.
Il punto di vista nella realizzazione del class diagram deve essere lo stesso che è stato usato negli use case cioè la realizzazione della gestione delle pagine dei docenti. Si procede con il classico approccio Bottom-Up per la realizzazione degli schemi. Per completezza e per inquadrare il problema, si riporta il diagramma delle classi relativo alla corretta gerarchia esistente fra le varie tipologie del personale presente nella facoltà.

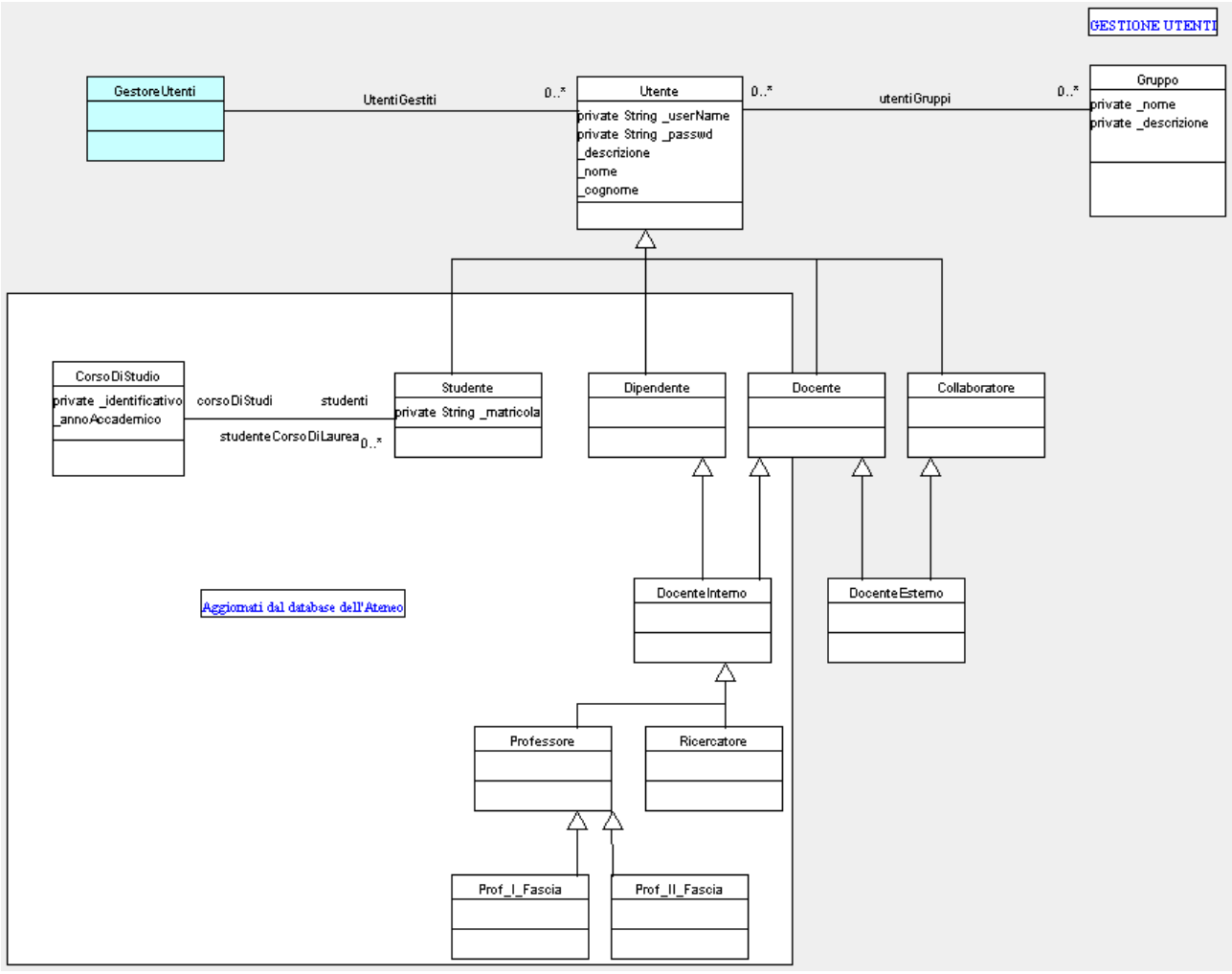


Figura 2.16: Class Diagram di generalizzazi one dei docenti

Come si può notare la porzione di class diagram realizzata in figura mette in evidenza le giuste dipendenze che esistono nel personale di facoltà. Lo schema delle classi realizzato per l'applicazione web allo stato attuale dovrà mantenere in qualche modo la *compatibilità* con quello proposto in figura 2.15.

Per motivi di facilità di realizzazione si è introdotta una classe docente che idealmente rappresentasse il collasso verso l'alto della struttura proposta. Anche se in seguito sarà necessario rivedere in parte questa scelta progettuale, una modifica in questa classe non avrà praticamente nessuna influenza sulla parte rimanente del progetto. Tale scelta è stata fatto tenendo conto anche del fatto che il linguaggio di programmazione scelto per la realizzazione del portale web non è in grado di utilizzare l'ereditarietà multipla, mentre nello schema proposto ad esempio "Docente Esterno" è una specializzazione delle due classi "Docente" e "Collaboratore", quindi questa particolare situazione sarebbe difficile da modellare.

Anche se in futuro verrà realizzata una area di accesso al sito generalizzata ed in grado di discriminare tra i gruppi di appartenenza degli utenti, per realizzare in pratica l'applicazione è stato necessario progettare una seppur più semplice, ma funzionale classe di accesso. Questa allo stato attuale è in grado di discriminare sulla base dello username e password, solamente due categorie di utenti, l'amministratore e il docente.

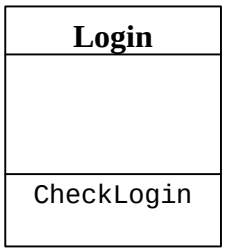


Figura 2.17: Classe Login

Il metodo pubblico CheckLogin verifica che effettivamente venga trovata una corrispondenza tra lo username e la password digitati e quelle memorizzate nel database. Se tale corrispondenza esiste l'accesso viene garantito altrimenti l'accesso viene negato.

La classe chiave della applicazione realizzata è quella dei docenti, tale classe comprende numerosi campi per la memorizzazione delle informazioni personali di interesse come i dati anagrafici, oppure informazioni più specifiche ad esempio l'ambito di ricerca. Vi sono poi alcuni metodi di carattere implementativo come Insert,Update,Delete che eseguono rispettivamente il salvataggio delle informazioni in una tabella del database.

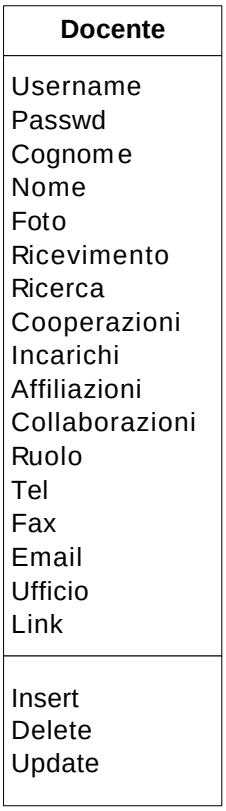


Figura 2.18: Classe dei Docenti

Analogamente per gli insegnamenti vengono descritti i campi più importanti, come specificato nel progetto del sito web di facoltà, questi sono il nome dell'insegnamento ,una descrizione testuale dell'insegnamento, e il relativo programma.

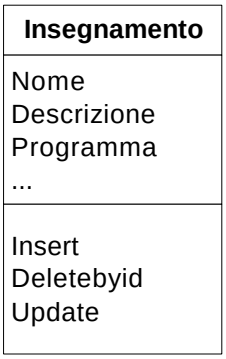


Figura 2.19: Classe degli insegnamenti

Per ogni insegnamento è possibile specificare la data degli appelli, gli avvisi e il materiale didattico.

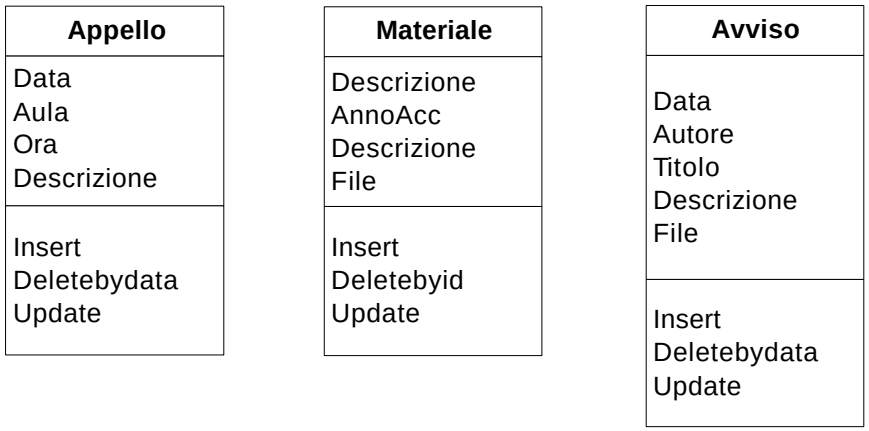


Figura 2.20: Classe degli Appelli, Materiale Didattico e Avvisi

Il diagramma completo delle classi per la logica di funzionamento della applicazione web realizzata è rappresentato in figura:

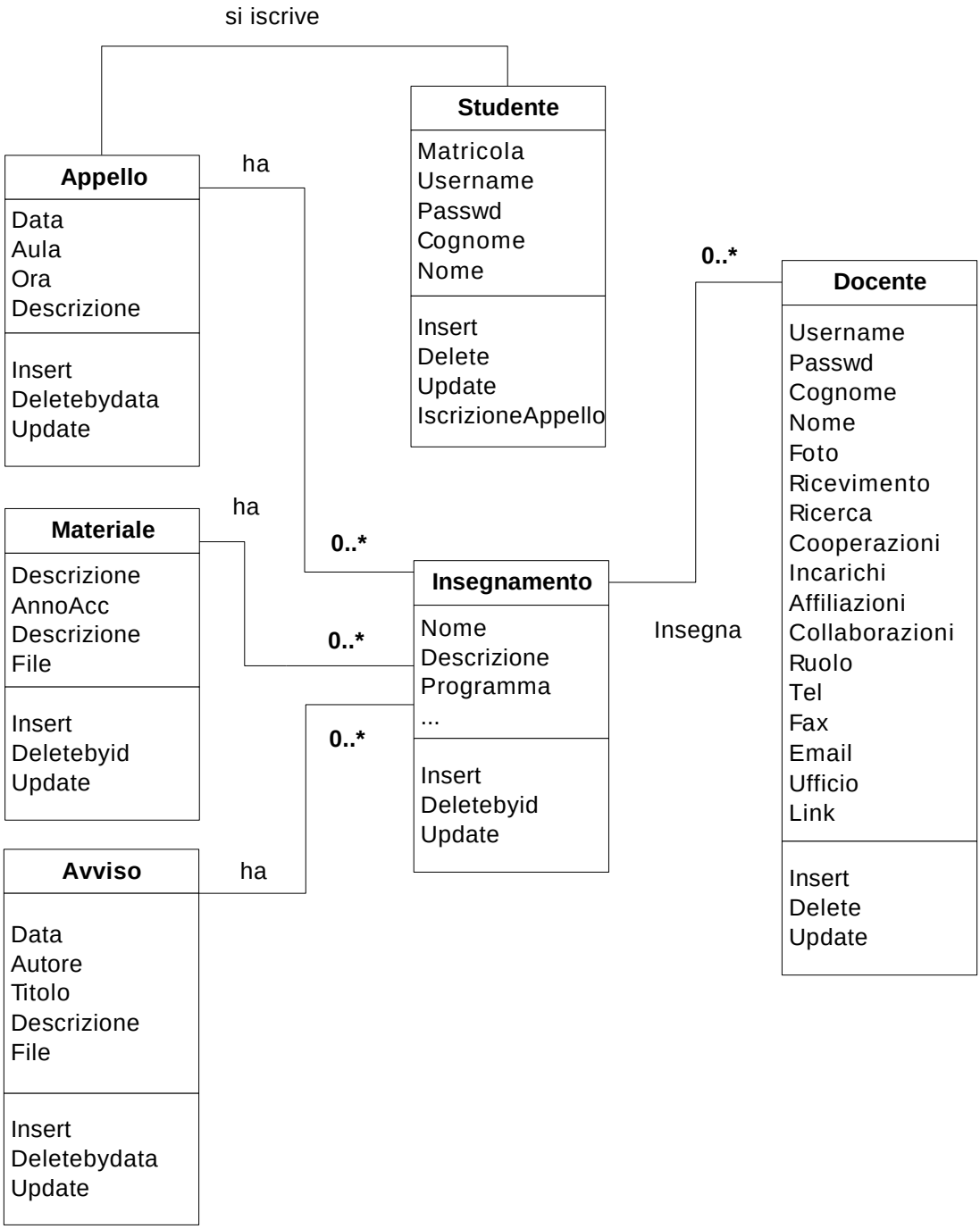
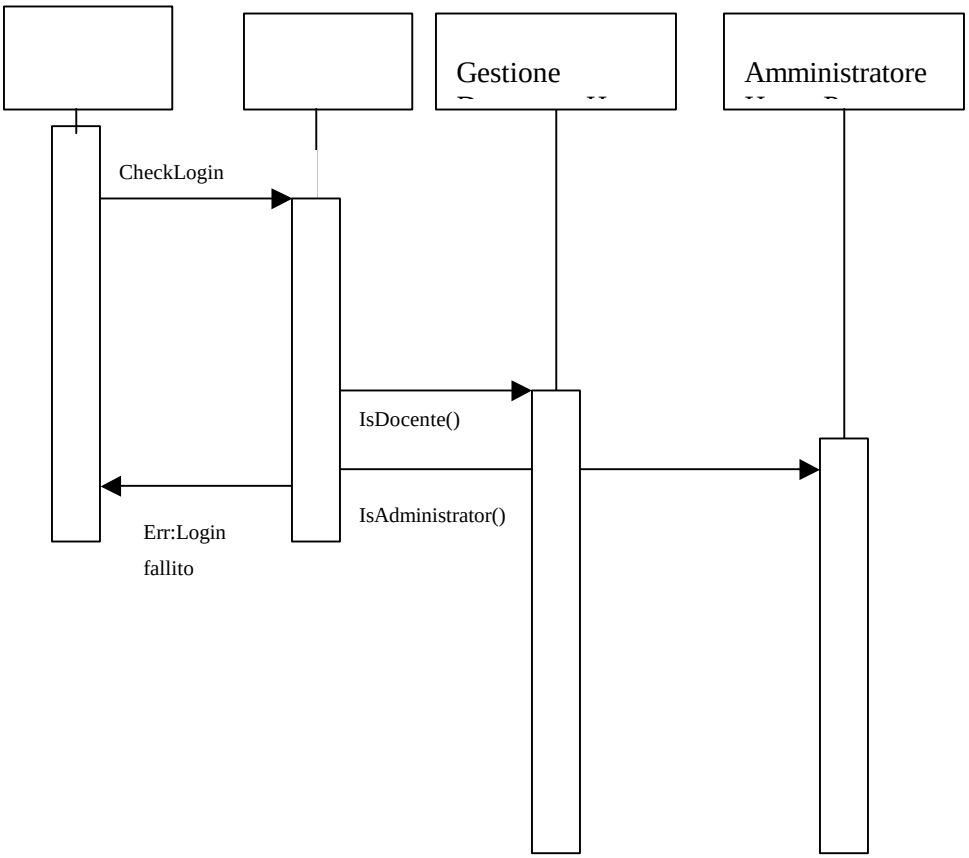


Figura 2.21: Diagramma delle classi della applicazione

2.3.3 SEQUENCE DIAGRAM

Attraverso il sequence diagram di UML si può modellare il funzionamento temporale della applicazione. In particolare può essere utile descrivere cosa avviene durante la fase di login di un utente nell'area di amministrazione del sito web.



Descrizione

L'utente fornisce in una apposita pagina l'username e la password., la cui validità viene controllata attraverso la classe User.

Se non viene trovato un match, il login fallisce e viene

Figura 2.22: Sequence diagram dell'area di login

Capitolo 3
Tecnologie di sviluppo

Questo capitolo ha l'intento di fornire informazioni di base sulle tecnologie software necessarie per la realizzazione di un sito web dinamico. In primo luogo verrà trattato il concetto di Web Server, fino alla spiegazione di cos'è un Application Server e all'influenza di un ambiente di sviluppo rapido nella realizzazione di un progetto software per le applicazioni web.

3.1 Il Web Server Apache

Un Web Server è un software in grado di ascoltare e soddisfare le richieste di un client, detto browser. Quando un Web Server riceve una richiesta, risponde mandando al mittente alcuni dati formattati in una pagina web che tipicamente contiene testo. Il protocollo sfruttato per la questa particolare comunicazione client/server è il Hypertext Transfer Protocol (HTTP). La maggior parte dei documenti richiesti sono realizzati in un linguaggio detto "Hypertext Markup Language" (HTML) che è un sottoinsieme di un altro linguaggio di markup detto "Standard General Markup Language" (SGML). Per inciso anche XML è un sottoinsieme di SGML.

In questo contesto verrà analizzato il web server Apache di Apache Software Foundation.
Questo Web Server è Open Source e completamente gratuito. Attualmente circa il 60% dei Web Server su Internet utilizza Apache, soppiantando altri prodotti commerciali come il famoso Internet Information Server (IIS) di Microsoft. La diffusione di Apache è da ricercarsi principalmente nelle sue performance, nella sua robustezza e nell'elevatissimo numero di moduli aggiuntivi che integrano le capacità del server.
Inoltre Apache è multi piattaforma, disponibile cioè per un grande numero di sistemi operativi differenti, basati su architetture hardware differenti (si va da Microsoft Windows, fino a Solaris di Sun Microsystem, ecc.). Ovviamente le prestazioni di Apache dipendono molto anche dal sistema operativo utilizzato, tipicamente è preferibile utilizzare un sistema operativo Unix-like, per ragioni di sicurezza e stabilità.

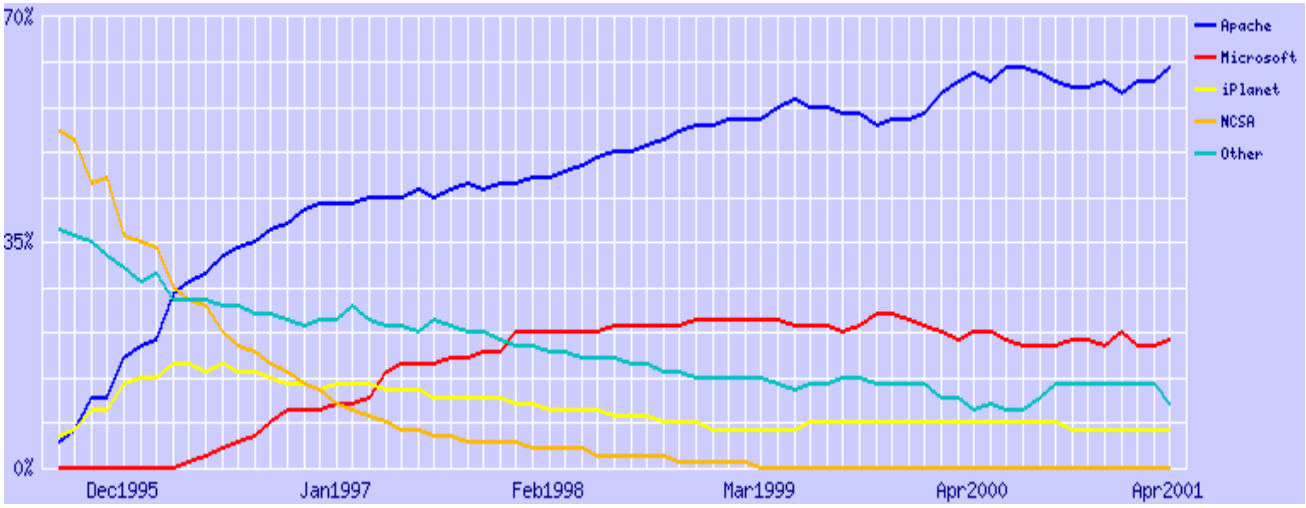


Figura 3.1: Diffusione di Apache

La figura 3.1 mostra l'andamento temporale dell'utilizzo di Apache rispetto ad altri Web Server (rif. Netcraft http://www.netcraft.co.uk Aprile 2001).

3.1.1 Caratteristiche Principali

Si analizzano brevemente alcune delle caratteristiche principali di Apache:

?? Apache è compatibile con le specifiche del protocollo HTTP/1.1
Conseguentemente permette la realizzazione dei cosiddetti Virtual Server. In altre parole è possibile "ospitare" più di un sito web sullo stesso server fisico, e farli funzionare correttamente e contemporaneamente tramite Apache. Naturalmente affinché Apache fornisca le pagine in modo corretto è necessario che anche il web browser supporti lo standard HTTP/1.1, comunque tutti i browser di recente realizzazione (dal 1996 in poi) supportano questa versione del protocollo.

?? E' possibile personalizzare gli errori del server attraverso l'uso di file o script. Quando il server intercetta un errore, mostra il file specificato o esegue un CGI al fine di permettere una diagnosi del problema on-the-fly. Questo consente al server di mostrare un messaggio di errore che sia comprensibile al visitatore o di eseguire qualche azione in risposta alla condizione di errore.

?? Una caratteristica molto apprezzabile quando si ha a che fare con siti multi lingua, è la capacità del server di riconoscere la lingua del browser e fornire a questo, se è disponibile,

un documento nella lingua corrispondente. In altre parole è possibile configurare Apache in modo da scegliere tra un insieme di documenti equivalenti, e di mandare al browser il più appropriato.

?? Apache è in grado di creare automaticamente una indice di file per ogni richiesta di un browser che punta su una specifica directory.

?? L'autentificazione degli utenti può essere realizzata in diversi modi, o con file indicizzati o con database relazionali. Sono poi disponibili moduli aggiuntivi per migliorare il livello di sicurezza fornito dal server.

Un altro punto a favore di Apache, è che la sua incredibile diffusione fa sì che la documentazione che riguarda questo software sia vastissima e di conseguentemente le esperienze dei sistemisti possono essere facilmente consultate per una facile messa a punto per quanto riguarda prestazioni e sicurezza.
Apache è un ottimo server, ma fornisce solamente pagine statiche, la necessità di realizzare applicazioni web invece, è dettata dal fatto che l'utente possa interagire con una seppur semplice interfaccia e modificare i dati visualizzati. Dunque è necessario avere a disposizione un ulteriore "strato" software che sia in grado di risolvere le richieste fatte da un browser, richieste che Apache non è in grado di soddisfare.

3.2 Tomcat Application Server

La necessità di realizzare pagine dinamiche, ha portato alla diffusione di numerosi Application Server di buone prestazioni. Tra questi si deve citare Tomcat del progetto Jakarta, una sezione dell'Apache Software Foundation. Tomcat è un ottimo Application Server che supporta lo standard Servlet 2.2 e Java Server Page (JSP). E' interamente scritto in Java ed è disponibile per numerose piattaforme inoltre, come il web server Apache è Open Source. Essendo stato sviluppato da una sezione del progetto Apache, risulta particolarmente facile l'integrazione con il Web Server.

Tomcat può anche funzionare come Web Server, ma è sconsigliabile l'utilizzo in ambienti al di fuori di quello di sviluppo, in quanto le prestazioni e l'affidabilità non sono assolutamente paragonabili a quelle di Apache. Di conseguenza l'utilizzo di Tomcat congiuntamente ad Apache fa

interazione tra il Web Server e l'Application Server utilizzando il formalismo dei casi d'uso di UML.

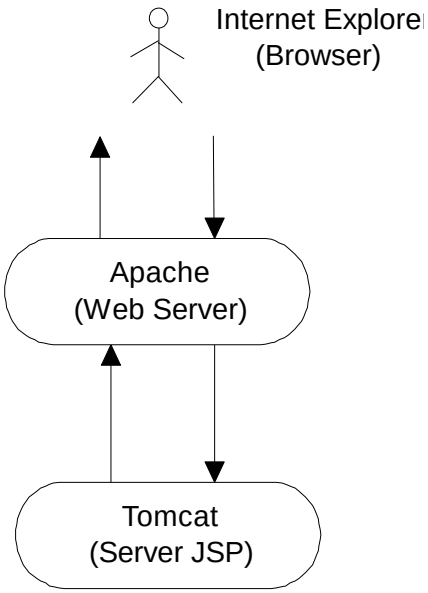


Figura 3.2: Interazione Browser- Apache-Tomcat

La realizzazione di una struttura a più livelli permette in generale un miglioramento della sicurezza del sistema e un performance tunig, più facile da realizzare. Capita molto spesso, infatti, che in sistemi reali esistano più server ognuno specificatamente configurato per eseguire un determinato compito. In questo caso è possibile che il Web Server e l'Application Server siano installati su due sistemi differenti e vengano fatti comunicare tra loro. Nel caso si renda necessario potenziare il sistema, soprattutto in termini di velocità, risulta così possibile determinare qual è il computer con il maggior carico di lavoro, e agire sul software o sull'hardware di quel sistema per migliorarne le prestazioni. Inoltre una struttura basata su una architettura costituita da più server risulta in generale più sicura rispetto ad una costituita da un solo server che realizza i servizi web e applicazioni.
Naturalmente gli svantaggi di una struttura di questo tipo sono in primo luogo di carattere economico, in quanto in generale due server hanno un costo superiore ad uno, in secondo luogo il tempo necessario per la configurazione e il mantenimento di più calcolatori risulta senz'altro più lungo rispetto a quello per un solo computer. La scelta deve essere fatta sulla necessità reale del sistema sulla base del carico di lavoro del server.

3.2.1 Installazione e configurazione

La configurazione dell'Application Server Tomcat, non è particolarmente difficile da realizzare. Di seguito, si farà riferimento alla versione di Tomcat 3.2.1, quella utilizzata per testare l'applicazione realizzata per il portale web di facoltà. Il sistema operativo utilizzato è Unix e per tale motivo la descrizione dell'installazione sarà fatta in conformità a questo sistema.
Si supponga di essere in possesso dei file di installazione di Tomcat, tipicamente il formato di questo file è un tarball (estensione **.tar.gz** oppure **.tgz)**. Prima di tutto è necessario decidere dove installare l'Application Server, un'ottima scelta è la posizione nel file system "/usr/local".
Tuttavia per l'installazione in questo percorso è necessario possedere i diritti di root, oppure contattare l'amministratore di sistema. Supposto di possedere i diritti di root, una volta effettuato il login si può procedere nel seguente modo.

```
# cd /usr/local
# tar xvfz tomcat-3.2.1.tar.gz
```

Se l'esecuzione del programma tar, va a buon fine nella directory /usr/local dovrebbe essere presente una sottodirectory chiamata tomcat-3.2.1 (o simile). La gerarchia di directory di Tomcat è costituita come segue a partire dalla home di Tomcat:

```
/bin      #contiene gli script per l'esecuzione del programma
/conf     #contiene i principali file di configurazione
/lib      #contiene le librerie necessarie per il funzionamento
/logs     #contiene i log di sistema
/webapps  #contiene applicazioni di esempio
/work     #contiene lo stato di esecuzione delle applicazioni
```

Per poter procedere alla esecuzione della applicazione potrebbe essere necessario configurare alcune variabili di ambiente. Supponendo di utilizzare la shell dei comandi "bash" si può procedere come segue:

```
# TOMCAT_HOME=/usr/local/tomcat-3.2.1
```

```
# export TOMCAT_HOME
# JAVA_HOME=/usr/local/java
# export JAVA_HOME
```

Una volta configurate correttamente le variabili di ambiente, l'Application Server è in grado di funzionare. Non è necessario, infatti, modificare il file di configurazione per effettuare un semplice test dell'applicazione, l'unica cosa importante è accertarsi che la porta 8080 sia disponibile. E' possibile utilizzare lo script "*startup.sh*" contenuto nella directory *bin* di Tomcat per eseguire l'Application Server. Se tutto va a buon fine aprendo un browser web all'url *http://localhost:8080/* si dovrebbe vedere una finestra simile a quella di figura.

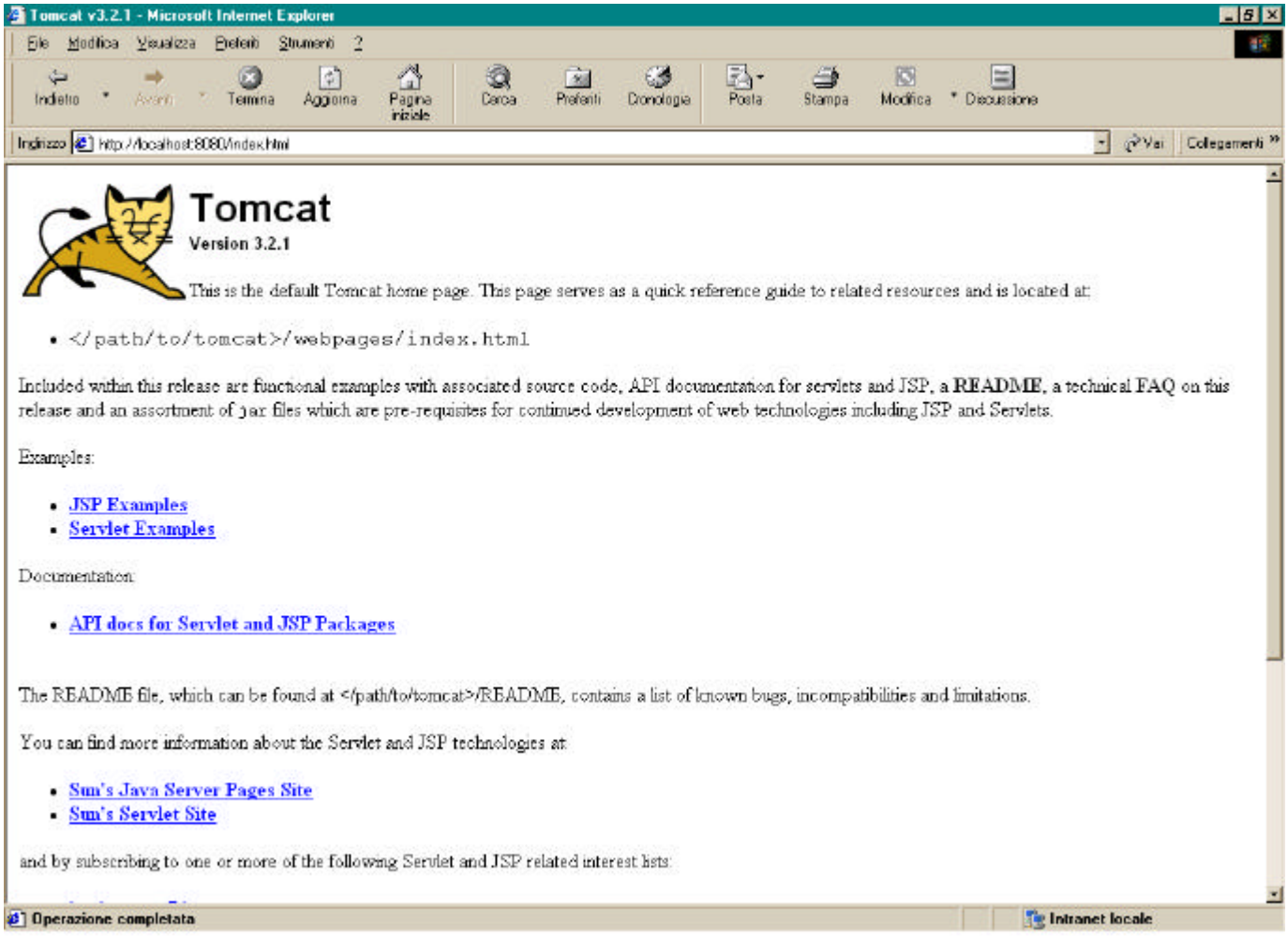


Figura 3.3: Test di funzionamento di Tomcat

Ciò garantisce che il Web Server contenuto in Tomcat funzioni correttamente, per testare il funzionamento di servlet e Java Server Pages è sufficiente seguire i relativi link. Per terminare l'esecuzione dell'Application Server è sufficiente eseguire lo script "*shutdown.sh*" contenuto sempre nella directory *bin.* In questa directory sono inoltre presenti anche i file batch necessari all'avvio di Tomcat in ambiente Microsoft Windows (sono *startup.bat* e *shutdown.bat*).

La configurazione di Tomcat, è realizzata attraverso un file XML, contenuto nella directory *conf*, chiamato *server.xml.* Il file di configurazione è bene commentato ed essendo scritto in XML permette una facile comprensione del funzionamento. Senza volere entrare completamente nel merito della descrizione del file *server.xml*, si evidenziano gli aspetti più significativi.

Innanzitutto, se la porta 8080 del sistema è già occupata, è possibile configurare Tomcat in modo da ascoltare le richieste su una porta differente. In questo caso è sufficiente modificare il campo *port* nella sezione *Connector* del file di configurazione:

```
<!-- Normal HTTP -->
<Connector className="org.apache.tomcat.service.PoolTcpConnector">
        <Parameter name="handler"
             value="org.apache.tomcat.service.http.HttpConnectionHandler"/>
        <Parameter name="port"
                value="8080"/>
</Connector>
```

In secondo luogo può risultare necessario specificare in quale percorso sono presenti le proprie servlet e jsp. In questo caso è necessario variare il valore di *docBase* nella sezione *Context* del file di configurazione:

```
<Context path="/examples"
                docBase="webapps/examples"
                crossContext="false"
                debug="0"
                reloadable="true" >
</Context>
```

In questo caso Tomcat "carica" all'url *http://localhost:8080/examples* le servlet e jsp che si trovano nel percorso relativo *webapps/examples* a partire dalla home directory dell'Application Server.

Analizziamo in dettaglio ogni campo della sezione *Context*.

- **Path.** Definisce il context path della applicazione cioè il prefisso della URI che viene richiesta. In questo caso Tomcat "carica" all'url *http://localhost:8080/examples*. Questo attributo è obbligatorio e deve iniziare con un carattere di slash ('/').
- **docBase**. E' la document root per l'applicazione web. Può essere un percorso relativo rispetto alla directory in cui è installato Tomcat oppure un percorso assoluto. Nell'esempio mostrato è un percorso relativo: `webapps/examples`. Questo attributo è obbligatorio.
- **debug**. Descrive il livello di dettaglio per i messaggi di log di Tomcat, quando l'applicazione è inizializzata, eseguita e arrestata. Il valore di default è zero e si riferisce ad funzioni di log essenziali, ma può essere specificato un valore diverso (0 – 9)
- **reloadable**. Permette a Tomcat di ricaricare al volo una servlet che è stata modificata, nella versione di Tomcat 3.2.1, questa funzione è ancora sperimentale, non sempre dà i risultati voluti, inoltre rallenta moltissimo il funzionamento dell'Application Server, e per tale motivo è sconsigliato usarla in applicazioni reali. E' tuttavia molto comoda durante la fase di sviluppo perché evita di riavviare il server dopo ogni cambiamento dell'applicazione. Può assumere i valori di true e false.
- **trusted**. Se impostata a true permette alla applicazione di avere accesso alle classi che costituiscono l'application server.

3.2.2 Cooperazione con Apache

Come già osservato in precedenza, Tomcat è un ottimo Application Server, ma non un buon Web Server. Nonostante abbia integrato in sé un semplice Web Server è molto meglio appoggiarsi a programmi specifici che offrano questo servizio, come Apache. Questo non vuole dire, che la funzione di Web Server di Tomcat sia inutile. E' sufficiente infatti, pensare ad una normale Workstation, che deve essere utilizzata per lo sviluppo di applicazioni web. Se Tomcat non avesse il servizio web, sarebbe necessario installare sulla Workstation anche Apache, appesantendo notevolmente il computer.
Inoltre i web server hanno caratteristiche di logs, configurabilità e prestazioni che superano notevolmente quelle di Tomcat, risulta quindi necessario instaurare una cooperazione tra il Web Server e l'Application Server, in modo tale da fare gestire le pagine statiche al Web Server, e quelle dinamiche all'Application Server, per avere una situazione ottimale. In queste circostanze verrà considerato il caso più semplice, Application Server e Web Server sono eseguiti sullo stesso computer.

Innanzitutto è necessario installare e configurare in Apache un modulo chiamato Jserv, per relative informazioni sulla installazione di questa libreria si rimanda a lla documentazione presente nel sito web *http://java.apache.org* . Supposto che l'installazione di questo modulo sia avvenuta correttamente è possibile procedere alla configurazione di Apache. Durante l'esecuzione di Tomcat viene generato dinamicamente nella directory *conf* un file con il nome di *tomcat-apache.conf* . Questo file in linea di massima contiene tutte le informazioni necessarie alla configurazione della cooperazione tra i due server. Nella maggior parte dei casi infatti è sufficiente includere questo file di testo nello script di configurazione di Apache che si trova in generale sotto la directory */etc* .

La sintassi del file sono analoghe a quelle del file di configurazione di Apache.

```
############################################################ #########
#      A minimalistic Apache-Tomcat Configuration File    #
#############################################################

# Note: this file should be appended or included into your httpd.conf

# (1) Loading the jserv module that serves as Tomcat's  apache adapter.
LoadModule jserv_module libexec/mod_jserv.so

# (1a) Module dependent configuration.
<IfModule mod_jserv.c>

# (2) Meaning, Apache will not try to start Tomcat.
ApJServManual on
# (2a) Meaning, secure communication is off
ApJServSecretKey DISABLED
# (2b) Meaning, when virtual hosts are used, copy the mount
# points from the base server
ApJServMountCopy on
# (2c) Log level for the jserv module.
ApJServLogLevel notice

# (3) Meaning, the default communication protocol is ajpv12
ApJServDefaultProtocol ajpv12
# (3a) Default location for the Tomcat connectors.
# Located on the same host and on port 8007
ApJServDefaultHost localhost
ApJServDefaultPort 8007

# (4)
ApJServMount /examples /root
# Full URL mount
# ApJServMount /examples ajpv12://hostn ame:port/root
</IfModule>
```

Una volta che la cooperazione tra Apache e Tomcat è stabilita, è sufficiente richiedere una pagina dinamica, senza specificare la porta di funzionamento di Tomcat. Ad esempio, supponendo di dover eseguire una servlet che visualizza un elenco di news, l'indirizzo corretto sarà:

http://localhost/servlet/NewsPage

In precedenza con l'uso del solo application server, era necessario specificare il seguente indirizzo:

http://localhost:8080/servlet/NewsPage

supposto ovviamente che questo sia in ascolto sulla porta 8080.

3.3.3 Sviluppo di applicazioni

Per realizzare servlet e JSP in modo efficiente è necessario seguire lo standard per lo sviluppo di applicazioni di questo tipo. In primo luogo quando si affronta lo sviluppo di un progetto di una certa entità, bisogna poter organizzare il codice ad esempio in sottodirectory. In un'applicazione web, la gerarchia di directory che si deve seguire prende il nome di "*Standard Directory Layout*".

- ***.html, *.jsp, etc.** - I file HTML e le pagine JSP, e tutti gli altri file che devono essere visibili ai browser compresi i file JavaScript e i fogli di stile possono essere contenuti in una singola directory. Nei progetti più grandi è possibile dividere questi tipi di file in directory differenti, ma per semplicità di manutenzione è spesso sufficiente contenere tutti i file in una sola directory.

- **WEB-INF/web.xml** - E' il *Web Application Deployment Descriptor* della applicazione. E' un file XML che descrive le servlet e i parametri di inizializzazione di queste ultime. E' anche chiamato *Web Container*, ed è di importanza fondamentale per il corretto funzionamento della applicazione.

- **WEB-INF/classes/** - Questa directory contiene ogni file di classe java e le relative risorse necessarie per la applicazione, incluso servlet e altri file di classe non uniti in java archivi (JAR). Se le classi sono organizzate in Java Packages, la struttura deve riflettere la gerarchia a partire dalla directory *WEB-INF/classes*.

- **WEB-INF/lib/** - Questa directory contiene file JAR e tutte le librerie necessarie per il funzionamento della applicazione.

- **WEB-INF/src/** - Questa directory contiene ogni file sorgente di tipo java.

Com'è stato detto il web container è d'importanza fondamentale per lo sviluppo di un'applicazione web. Il file è strutturato in modo molto semplice, ed è un file XML perciò ha una sintassi ben precisa e di facile comprensione. Il file *web.xml* contiene i parametri di

inizializzazione d'ogni servlet, e ogni servlet deve essere specificata all'interno del file altrimenti non può funzionare.

Di seguito viene presentata una parte del file a titolo di esempio:

```
<!-- PAGINA DEGLI INSEGNAMENTI DEI DOCENTI  -->

<servlet>
  <servlet-name>  DocInsPage </servlet-name>
  <servlet-class> DocInsPage </servlet-class>

  <init-param>
     <param-name> url </param-name>
     <param-value>jdbc:odbc:Ing</param-value>
  </init-param>

  <init-param>
     <param-name>  user   </param-name>
     <param-value> Administrator </param-value>
  </init-param>

  <init-param>
     <param-name>  passwd    </param-name>
     <param-value>           </param-value>
  </init-param>

  <init-param>
     <param-name>     driver                              </param-name>
     <param-value>    sun.jdbc.odbc.JdbcOdbcDriver        </param-value>
  </init-param>

</servlet>
```

Per ogni servlet è possibile specificare in nome della classe e il nome che deve essere mostrato nell'url (nell'esempio questi coincidono). Per ogni servlet è possibile specificare un qualsiasi numero di parametri iniziali purché questi abbiano un nome differente. Ogni parametro deve avere un nome e un valore, che può anche essere nullo. Nell'esempio mostrato in figura vengono passati ad una servlet di nome DocInsPage le informazioni di connessione relative ad un generico database, la connessione viene effettuata tramite i driver JDBC. E' possibile recuperare le informazioni contenute nel web container dalla servlet con alcune semplici istruzioni:

```
//Acquisizione dal web container
  url=getServletConfig().getInitParameter("url");
  user=getServletConfig().getInitParameter("user");
  passwd=getServletConfig().getInitParameter("passwd");
  driverName=getServletConfig().getInitParameter("driver");
```

I parametri passati sono tutti del tipo String.

Adesso dovrebbe essere evidente l'importanza del web container. Infatti, è possibile modificare solamente i valori del web container senza ricompilare la servlet per ottenere sostanziali modifiche, nell'esempio proposto è addirittura possibile cambiare completamente le informazioni di connessione relative ad un database, variando username, password, e driver JDBC. Naturalmente è possibile specificare qualsiasi altro tipo di parametro, non necessariamente le informazioni di connessione ad un database.

3.3 Enhydra e XMLC

Con il diffondersi di Internet e di linee dati ad alta velocità, le applicazioni web stanno prendendo sempre più piede. In questo particolare contesto si stanno delineando diverse tecnologie per lo sviluppo di tali software. Tuttavia, a causa del forte contenuto grafico, una applicazione web coinvolge sempre più di una figura professionale. Infatti, oltre ai programmatori, tecnici e analisti, vengono sempre coinvolti nello sviluppo, grafici, web designer e artisti. Come è immediato supporre le conoscenze tra le differenti tipologie di figure che lavorano su una applicazione di questo tipo sono moltissime. Per questo, nel panorama mondiale, si stanno sviluppando diverse metodologie per consentire la cooperazione tra queste differenti figure professionali, una tra questa è Enhydra Application Server e il suo tool XMLC.
Il problema non è di facile soluzione perché gli strumenti utilizzati per lo sviluppo di ogni aspetto dell'applicazione web possono essere tra i più disparati: il personale tecnico e di programmazione può usare ambienti di sviluppo come RAD, ha una ottima conoscenza di linguaggi di programmazione ecc., il grafico utilizza prevalentemente software di fotoritocco, oppure programmi di web editing come FrontPage o DreamWeaver oppure Flash.

Tuttavia una qualsiasi applicazione web moderna nel suo ciclo di vita coinvolge prima o più entrambe le figure professionali. L'esigenza immediata è dunque quella di consentire ad entrambe le parti di lavorare sullo stesso progetto senza eccessive interferenze.
L'obiettivo ideale è quello di tenere separata la presentazione dalla business logic della applicazione web. A differenza di un qualsiasi altro software, una pagina web può subire mutamenti nell'aspetto grafico con frequenza anche settimanale (necessità dettata ad esempio dal cambiamento della moda o per motivi pubblicitari), sarebbe dunque impensabile dovere ritoccare diverse centinaia di linee di codice per mutare l'aspetto grafico di una pagina realizzata ad esempio solo tramite l'uso di servlet java oppure tramite CGI. Uno dei problemi fondamentali è dunque quello della facilità di manutenzione e richiede una netta separazione tra:

- Presentazione della pagina (Presentation)
- Logica di funzionamento (Business Logic)

3.3.1 Enhydra Application Server

Al fine di rendere più agevole lo sviluppo di una applicazione web, molti produttori di software hanno iniziato a raccogliere librerie (spesso utilizzando le risorse messe a disposizione dal mondo Open Source), al fine di realizzare Application Server di buone prestazioni e di facile utilizzo. Uno esempio particolare tra questi è Enhydra di Lutris Tecnology. Enhydra è realizzato in parte con il codice di una altro noto Application Server, Tomcat del progetto Jakarta di Apache. Entrambi sono in grado di gestire tecnologie come servlet, e java server page, ma Enhydra fornisce anche alcuni strumenti per la separazione tra la presentation logic e la business logic, facilitando molto lo sviluppo e la manutenzione di un sito web dinamico.

L'architettura di Enhydra è del tipo Multi-Tier come si può ben capire dallo schema rappresentato in figura:

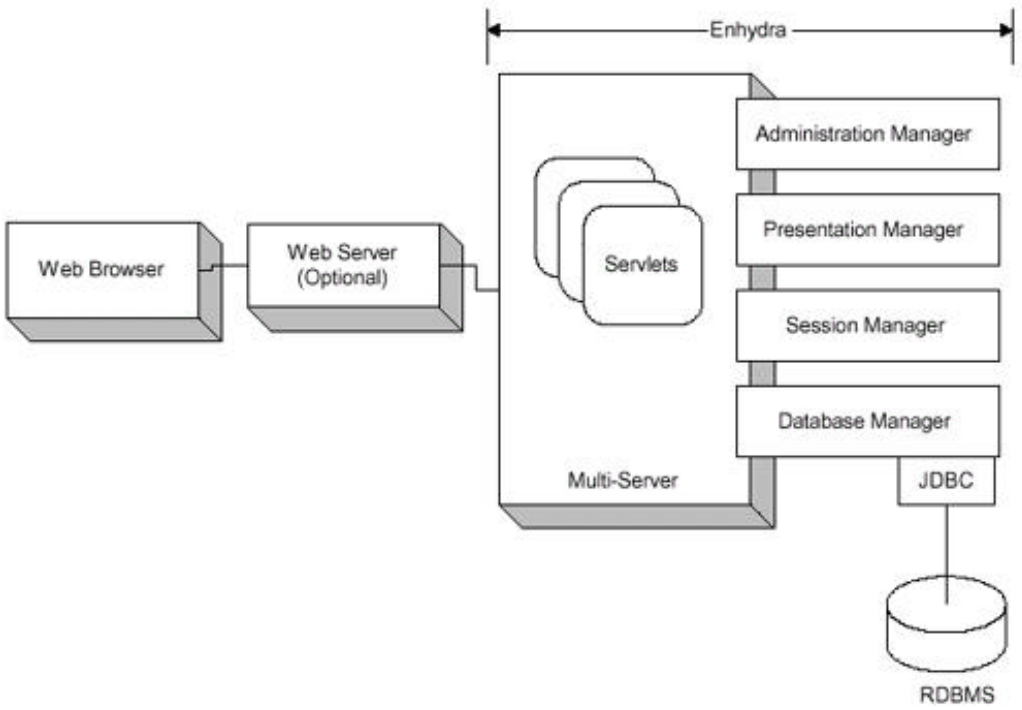


Figura 3.4: Architettura di Enhydra

Analizziamo in breve ogni suo componente.

Administration Manager

L'administration manager è un tool grafico che permette di co ntrollare il funzionamento del server e di effettuare le seguenti operazioni

- ?? Start/Stop di Web Application e servlet
- ?? Aggiungere/rimuovere applicazioni e servlet
- ?? Modificare i parametri
- ?? Verificare lo stato delle applicazioni (numero di connessioni…)

Nonostante Enhydra derivi in parte da Tomcat, quest'ultimo non ha a disposizione uno strumento così evoluto per l'amministrazione. Inoltre necessita di essere riavviato nel caso in cui sia necessario aggiungere una nuova applicazione e cioè implica una temporanea inattività del servizio

Session Manager

Com'è noto il protocollo di comunicazione tra un Web Browser e un Web Server è del tipo stateless (cioè senza stati), questo significa che il server non è a conoscenza dello stato in cui si trova il web browser e non è in grado di controllarne l'esecuzione. Un esempio invece di protocollo di comunicazione statefull è FTP, dove è necessario effettuare un login, e alla fine della sessione un logout, essendo il server FTP durante il periodo di collegamento a conoscenza dello stato del Client. L'esigenza di sopperire alla mancanza di stati nel protocollo di comunicazione http, ha creato la necessità di introdurre il concetto di sessione. Queste sono particolarmente utili quando è necessario tenere traccia delle operazioni eseguite dal client, nei casi di autentificazione d'utenti oppure nei siti di e-commerce per salvare lo stato del "carrello". In genere ad ogni utente, viene assegnato un numero identificativo univoco, e una durata della sessione. Il session manager di Enhydra si occupa di gestire le sessioni.

Database Manager

Nelle pagine web dinamiche, capita sovente di dover reperire informazioni da uno o più database. Il database manager di Enhydra mette a disposizione alcune classi Java che estendono quelle standard fornite dai driver JDBC facilitando l'esecuzione di query (attraverso l'oggetto DBQuery) e transazioni (con l'oggetto DBTransaction). Naturalmente il Database Manager può gestire un qualsiasi numero di connessioni attraverso vari database, come schematizzato in figura.

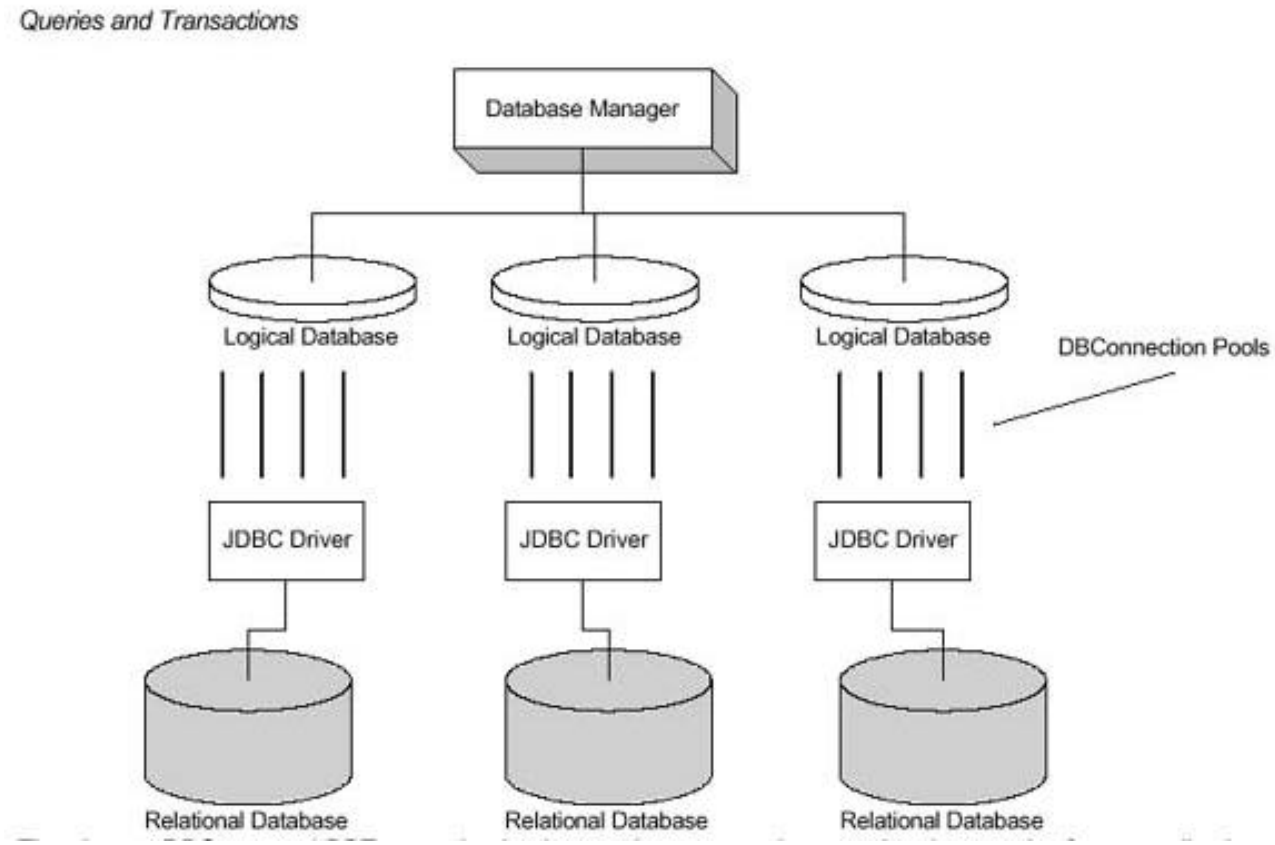


Figura 3.5: Architettura del Database Manager di Enhydra

3.3.2 XMLC

Una libreria di sicuro interesse contenuta in Enhydra Application Server è senza dubbio XMLC (cioè XML Compiler). La diffusione di questo pacchetto è stata molto bene accolta dagli sviluppatori di applicazioni web, tanto che, a richiesta del pubblico, è recentemente diventato un progetto stand alone per la società Lutris Tecnology. Inoltre, un recente sondaggio di una famosa rivista di programmazione lo pone tra le tecnologie più sfruttate nella realizzazione di applicazioni per il web.
L'obiettivo di XMLC è ottenere la separazione tra l'aspetto grafico di una pagina web dinamica e il codice di quest'ultima realizzato facendo uso di servlet java. Le servlet sono indubbiamente un ottimo metodo (sicuro e robusto) per realizzare pagine web dinamiche, ma hanno il limite di incorporare nel codice stesso anche l'aspetto grafico della pagina. Con XMLC è possibile realizzare delle servlet che contengono solamente la business logic (cioè la logica di funzionamento della applicazione) e delegare l'aspetto grafico ad un'altra figura professionale. Quest'ultima per realizzare il layout della pagina è libera di usare qualsiasi tool di web editing preferisca (in parte anche Flash di Macromedia).

Il funzionamento di XMLC è molto semplice, nonostante ciò molto efficace. Si basa sostanzialmente nella trasformazione di un documento XML o HTML (quindi anche XHTML) in una classe Java.
L'insieme di funzioni API (Application Program Interface) di XMLC consente di manipolare la struttura di un documento XML, che come è noto ha una natura duale. Infatti, si può affermare che un documento è realizzato in XML quando è un contenitore di informazioni che può essere descritto come:

- Una sequenza lineare di caratteri (con TAG)
- Una struttura dati astratta rappresentabile attraverso un albero.

Gli oggetti che mette a disposizione XMLC per la manipolazione della pagina agiscono proprio sulla struttura ad albero del documento XML. La struttura ad albero prende il nome di DOM (Document Object Model). E' possibile con questo metodo manipolare la pagina, aggiungendo tabelle, hyperlink, paragrafi, ecc.
XMLC, infatti, mette a disposizione alcuni metodi per avere accesso ai TAG contenuti nel documento che hanno un identificativo (id), consentendo di modificare, ad esempio, il testo di un paragrafo. Questo non solo rende più semplice manipolare i contenuti della pagina, ma fornisce anche un modo per verificare a livello di compilazione se la pagina è cambiata in modo tale da influire sul codice che genera la parte dinamica. Se un evento simile si verifica è individuato a run-time dal compilatore Java, e non solo da un'attenta analisi visuale della pagina dinamica. Un approccio di questo tipo consente dunque una maggiore sicurezza nello sviluppare applicazioni che funzionano correttamente.

Esempio di funzionamento

Per illustrare nel migliore dei modi il funzionamento di XMLC è sufficiente fare un esempio.
Si supponga di avere un file scritto in HTML, per convertirlo nella relativa classe java è sufficiente eseguire il comando:

```
$ xmlc –keep helloworld.html
```

Come risultato si ottengono due file:

- helloworld.java
- helloworld.class

Naturalmente tra i metodi disponibili vi è quello che consente di ricreare il documento originale (cioè in HTML) a partire da quello convertito (cioè la classe Java).

Si consideri ora la più semplice servlet possibile:

```
import java.io.* ;
import java.text.* ;
import java.util.* ;
import javax.servlet.* ;
import javax.servlet.http.* ;
```

```
public class HelloWorldExample extends HttpServlet {


    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        ResourceBundle rb =
            ResourceBundle.getBundle(“LocalStrings”,request.getLocale());
        response.setContentType(“text/html”);
        PrintWriter out = response.getWriter();

        out.println(“<html>”);
        out.println(“<head>”);
        out.println(“<title> Hello World </title>”);
        out.println(“</head>”);
        out.println(“<body bgcolor=\”white\”>”);
        out.println(“<body>”);
        out.println(“<h1>Hello World</h1>”);
        out.println(“</body>”);
        out.println(“</html>”);
    }
}
```

Questa servlet non fa altro che creare una pagina con la scritta “Hello World” nel documento HTML. Come si può notare all’interno del codice Java (nella parte scritta in grassetto) ci sono molte istruzioni che “disegnano” la pagina, quindi vi è un’evidente interferenza tra business logic e presentazione della pagina. In un simile contesto l’unica tipologia di persone che può lavorare su questa pagina è il programmatore, non certo il web artist.
Si supponga di delegare il lavoro di realizzazione della stessa pagina al web artist e di utilizzare XMLC. Una volta convertita la pagina HTML in una classe Java la servlet di cui sopra diventa:

```
import java.io.* ;
import java.text.* ;
import java.util.* ;
import javax.servlet.* ;
```

```
import javax.servlet.http.* ;

import helloworld.class.*;

public class HelloWorldExample extends HttpServlet {


    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        helloworld HT= new helloworld();
        ResourceBundle rb =
            ResourceBundle.getBundle(“LocalStrings”,request.getLocale());
        response.setContentType(“text/html”);
        PrintWriter out = response.getWriter();

        out.println(HW.toDocument());
    }
}
```

L’unica istruzione che si occupa della stampa del documento è：

```
out.println(HW.toDocument());
```

quindi vi è praticamente la completa separazione tra l’aspetto grafico e la applicazione vera e propria. Naturalmente l’esempio di cui sopra è molto semplice, le cose si complicano un po’ quando è necessario realizzare pagine più complesse, ma la separazione tra presentazione e business logic, rimane comunque apprezzabile. Per mettere in evidenza come XMLC interagisca con la struttura DOM di un documento, si consideri il seguente caso. Supponiamo di volere estendere il contenuto di una generica tabella con nuove righe (situazioni analoghe capitano sovente in pratica). La struttura DOM di una tabella nella sua rappresentazione più generale la seguent e:

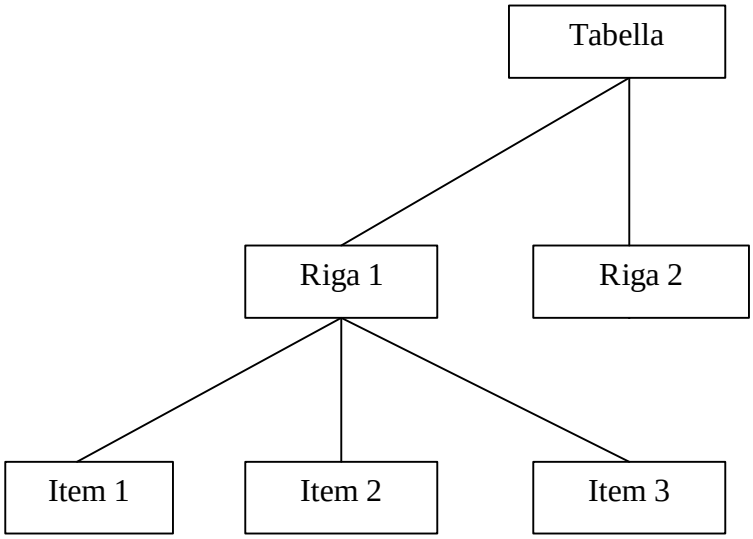


Figura 3.6: Struttura DOM di una tabella

Consideriamo ora questo breve listato di HTML che disegna una tabella con due righe e tre colonne in una pagina web:

```
<table id=”SampleTable”border=”1” width=”70%”>
  <tr id=”HeaderRow”>
    <td width=”33%”>Header1</td>
    <td width=”33%”>Hedear2</td>
    <td width=”34%”>Header3</td>
  </tr>
  <tr id=”SampleRow”>
     <td width=”33%”>Item1</td>
    <td width=”33%”>Item2</td>
    <td width=”34%”>Item3</td>
  </tr>
</table>
```

Eseguendo XMLC con lo switch –dump si ottiene anche la struttura DOM del documento. Il risultato del parsing dello stralcio di HTML di cui sopra è il seguente:

```
HTMLTableElementImpl: TABLE: id=”SampleTable”
   HTMLTableElementRowImpl: TR: id=”HeaderRow”
      HTMLTableCellElementImp: TD:
         LazyText: Header1
      HTMLTableCellElementImp: TD:
         LazyText: Header2
      HTMLTableCellElementImp: TD:
         LazyText: Header3
   HTMLTableElementRowImpl: TR: id=”SampleRow”
      HTMLTableCellElementImp: TD:
         LazyText: Item1
      HTMLTableCellElementImp: TD:
         LazyText: Item2
      HTMLTableCellElementImp: TD:
         LazyText: Item3
```

La gerarchia è data dall’indentazione del testo. HTMLTableElement, HTMLTableElementRow ecc, sono le implementazioni degli oggetti java della classe HTMLElement e, attraverso questi, si può manipolare la pagina HTML. Con riferimento alla tabella di prima, supponiamo di volere aggiungere un numero qualsiasi di righe, è sufficiente eseguire la seguente semplice interazione

```
SampleTable ST=new SampleTable();


HTMLTableElement table = ST.getElementSampleTa ble();
HTMLTableRowElement sampleRow = ST.getElementResultRow
for (j=0; j<100;j++)
 {
  HTMLTableRowElement dataRow =(HTMLTableRowElement)sampleRow.cloneNode(true);
  table.appendChild( dataRow );
 }
table.removeChild(sampleRow);
```

L’oggetto ST appartenete alla classe SampleTable è generato automaticamente da XMLC a partire dal documento HTML (o XML). Ovviamente all’interno del ciclo for è possibile ad esempio inserire in modo opportuno qualsiasi informazione associata alla nuova riga inserita. L’esempio svolto è un po’ più articolato, ma come si può notare, non ci si è assolutamente interessati degli

aspetti grafici della pagina, infatti tutte le informazioni riguardanti il layout della tabella vengono acquisite dalla tabella originale e clonate di riga in riga.

Ciclo di sviluppo con XMLC

Un pacchetto software di questo tipo indubbiamente influisce molto sul ciclo di sviluppo di una applicazione web. Nello sviluppo di un qualsiasi progetto software, vi sono numerosi incontri tra il committente dell'applicazione e il produttore della stessa, al fine di definire le specifiche dei requisiti. Nello sviluppo di un sito web dinamico, il committente spesso da molta più importanza all'aspetto grafico del sito piuttosto che alle reali funzionalità. Questo perché è molto facile avere una immagine di quello che si vuole ottenere, quindi la veste grafica ha senza dubbio un peso molto alto, e in genere è molto più facile e veloce da realizzare che scrivere centinaia di linee di codice di una applicazione. Per questo il lavoro iniziale dello sviluppo di un sito web è senza dubbio delegato ad un grafico, che realizzerà il layout di molte pagine. Questo può portare ad un senso di soddisfazione da parte cliente, che in breve tempo vede realizzata una parte di ciò che sarà la sua applicazione web.
Contemporaneamente può essere iniziato lo sviluppo della business logic della applicazione. L'aspetto grafico, nel frattempo, subirà notevoli mutamenti nel corso dello sviluppo del sito web, per adattarsi alle esigenze e ai gusti del committente ma, utilizzando XMLC, moltissime modifiche non avranno alcuna influenza sulla programmazione della applicazione web, consentendo dunque un lavoro parallelo tra coloro che si occupano della parte grafica e coloro che si occupano della logica di funzionamento. Ciò in linea di massima comporta un accorciamento dei tempi di realizzazione del progetto, con conseguente risparmio economico di entrambe le parti, la software house e il committente.

3.4 Forte for Java

La necessità di sviluppare applicazioni web ha fatto sì che numerose software house realizzassero i propri ambienti di sviluppo il cui obiettivo sia facilitare la realizzazione di queste applicazioni, di componenti e documentazione. Per tale motivo questi ambienti di sviluppo sono detti RAD cioè Rapid Application Development. A seconda della piattaforma e del linguaggio di programmazione ne esistono differenti:

- ?? Microsoft Visual Interdev in ambienti Windows
- ?? Borland Jbuilder in ambienti Linux e Windows
- ?? Forte for Java in ambienti Unix e Windows
- ?? WebSphere di IBM in ambienti Unix e Windows

“Forte for Java” è un ambiente di sviluppo RAD realizzato da Sun Microsystem, che consente di sviluppare efficacemente applicazioni web based. Forte for Java e gli altri tool utilizzano in genere un IDE (Integrated Development Environment) per facilitare la stesura del codice. Forte for Java è un ambiente di sviluppo completo ed efficiente in grado di realizzare semplici applicazioni di tipo web-centric (cioè web clients che accedono ad un singolo Application Server e accedono ad un database) oppure applicazioni di carattere enterprise distribuite.

Secondo la necessità, questo RAD permette di scrivere componenti JavaBeans, applets, Java Server Pages oppure servlet. Inoltre consente di lavorare ad un gruppo di sviluppatori sullo stesso progetto nel caso in cui si renda necessaria la collaborazione e la condivisione del codice.

Forte for Java è attualmente disponibile in due versioni

- ?? Community Edition
- ?? Internet Edition

La prima è freeware e manca di a lcune funzionalità, mentre la seconda è più completa, e permette di lavorare in team e implementa i servizi dell’Application Server Tomcat. Di seguito verrà considerata la versione Internet Edition.

Nei paragrafi precedenti è stato accennato alle applicazioni di tipo web centric, queste sono le più semplici applicazioni che si possono realizzare e dovrebbero essere utilizzate solamente in progetti molto semplici, perché non permettono nessun tipo di modularità.
Attraverso l'uso dell'Application Server di Forte for Java è possibile realizzare e testare il funzionamento di applicazioni di tipo web centric con le seguenti caratteristiche:

- Control and Sequencing
- Business Logic
- Persistence
- Presentation

Queste non sono le sole funzioni dell'Application Server, che deve naturalmente essere in grado di tenere traccia degli utenti e delle sessioni.

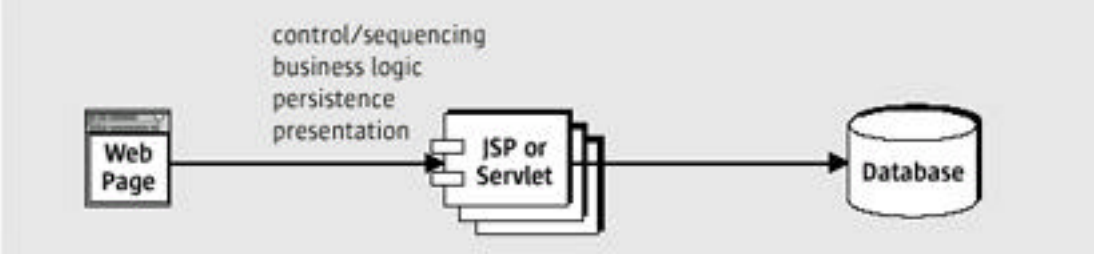


Figura 3.7: Approccio Web-Centric

In figura è possibile vedere una semplice applicazione web-centric.
Nel caso in cui sia necessario invece realizzare un'applicazione maggiormente modulare, dove funzioni differenti, sono realizzate da componenti differenti, si fa sentire maggiormente il peso di un buon RAD come Forte for Java. In questi casi, infatti, viene premiata la chiarezza del codice nei termini di riuso, leggibilità, e conformità alle specifiche del progetto, consentendo inoltre la separazione a livello di codice tra business logic e presentation. In genere in progetti di maggior entità, la concezione web-centric non è sufficiente, anzi non deve essere utilizzata, pena l'impossibilità o quasi di manutenzione del codice. Un approccio migliore è visibile in figura dove ad esempio la *presentation* viene realizzata attraverso l'uso di Java Server Page che sono supportate direttamente dall'IDE. Indipendentemente dalla necessità di seguire in modo stretto lo schema

proposto, viene mostrata come in realtà dovrebbe essere realizzata in modo corretto un'applicazione web di una certa complessità.

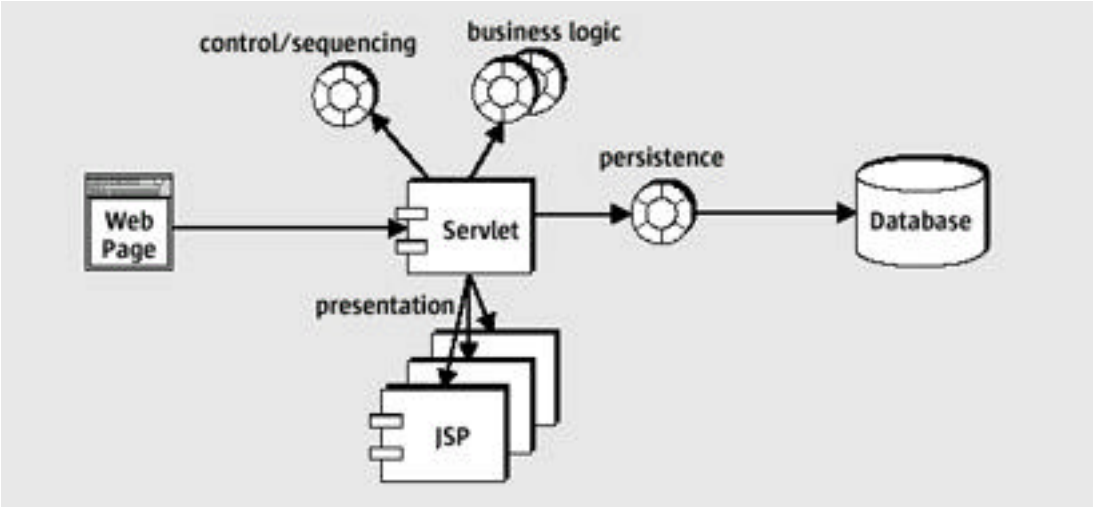


Figura 3.8: Approccio Modulare

La modularità, infatti, è una caratteristica molto importante nei progetti, altre sono le prestazioni l'affidabilità e la sicurezza.Queste caratteristiche sono spesso richieste da applicazioni che devono essere distribuite dove differenti componenti girano su diversi computer al fine di massimizzare le prestazioni, il load balancing, sicurezza e prevenire crash di sistema. Valutare queste caratteriste a livello di progetto aiuta senz'altro a stabilire le giuste tecnologie per lo sviluppo delle applicazioni.

3.4.1 Scegliere tra JSP e Servlet

L'ambiente di sviluppo analizzato consente di realizzare facilmente sia servlet che JSP, viene dunque da chiedersi, quando sia meglio utilizzare una o l'altra tecnologia. Senza voler avere la pretesa di essere esaustivi, di seguito vengono presentati alcuni concetti fondamentali che distinguono servlet e JSP.

Servlet

Le servlet sono usate per estendere le funzionalità di un web server consentendo si realizzare pagine dinamiche. Attraverso le servlet API un programmatore può accedere facilmente alle richieste (*requeste*) nel protocollo HTTP che vengono fatte da un browser, e generare "*response HTTP*" attraverso l'uso di oggetti Java, che forniscono molti metodi per manipolare queste richieste. Le servlet sono usate tipicamente dove è necessario ottenere informazioni in seguito alle richieste generate da un form HTML, spesso interrogando una sorgente di dati. Sono anche utilizzate per

controllare il flusso della applicazione web, consentendo ad esempio di abilitare o disabilitare l'accesso a certe risorse. Forse l'applicazione più comune realizzata attraverso l'uso di servlet è quella di tenere traccia della sessione degli utenti, tipico esempio è la realizzazione di un carrello in una applicazione di e-Commerce.

Java Server Pages

Una pagina JSP è un componente web basato principalmente su testo, che viene tradotto in una servlet prima della sua esecuzione. La prima grande differenza tra servlet e JSP è che queste ultime fanno parte di quelle tecniche di programmazione che prendono il nome di HTML embedded, mescolano cioè codice HTML con codice tipico di un linguaggio di programmazione (ad esempio Java per le JSP e VB-Script per le ASP).

Dalla prospettiva di un ipotetico utente infatti, una pagina JSP, ha una struttura completamente rovesciata rispetto ad una servlet. Infatti in una descrizione semplicistica si può definire una JSP come una pagina HTML che contiene codice Java, esattamente l'opposto di una servlet che è un programma scritto in Java che si occupa di stampare alcune linee di HTML.

Le fasi principali di funzionamento di una JSP, cioè il cosiddetto ciclo di vita è costituito da tre instanti temporali differenti:

- **Traduzione -** Questo processo si riferisce alla trasformazione di una JSP in una servlet la prima volta che viene fatta una richiesta della pagina. Le volte successive che viene richiesta la stessa pagina questo processo non viene eseguito.
- **Instanziamento -** Quando viene ricevuta una richiesta per una particolare pagina JSP, l'Application Server cerca di localizzare una istanza corrispondente, se non ne viene trovata, crea una nuova instanza.
- **Distruzione** - L'Application Server può recuperare alcune risorse di sistema "distruggendo" l'istanza di una pagina JSP. Normalmente viene fissata una durata temporale che indica quanto l'instanza di una JSP può persistere senza ricevere nessuna richiesta.

Come è stato visto la differenza sostanziale tra servlet e JSP è che le prime fanno parte di quelle tecniche di programmazione non HTML embedded, mentre le seconde fanno parte di quelle HTML embedded. Realizzare dunque pagine web ricche di contenuti grafici, tramite il solo uso di servlet, può risultare molto difficile. Per questo motivo spesso, solamente la logica della applicazione, viene realizzata attraverso questo tipo di tecnologia. E' possibile dividere una applicazione web in tre componenti fondamentali:

?? **Front Component** - Questo oggetto si occupa di ricevere le HTTP request dal web server ed è in grado di processarle oppure di ridirigerle ad un altro componente.

?? **Logic Component** - Questo tipo di oggetto effettua una qualsiasi operazione legata alla logica della applicazione che si deve realizzare. Per esempio può controllare il flusso dei dati nella applicazione, interrogare un database o processare la business logic. Questo componente è realizzato molto spesso da servlet o altre classi.

?? **Presentation Component** - Questo componente genera una parte o tutta la "HTTP response" da mandare al client browser. JSP sono la scelta migliore per questa funzione, ma per pagine molto semplici in termini di contenuto grafico e possibile utilizzare anche le servlet. Oppure sfruttare queste ultime insieme ad altre tecnologie: ad esempio XMLC.

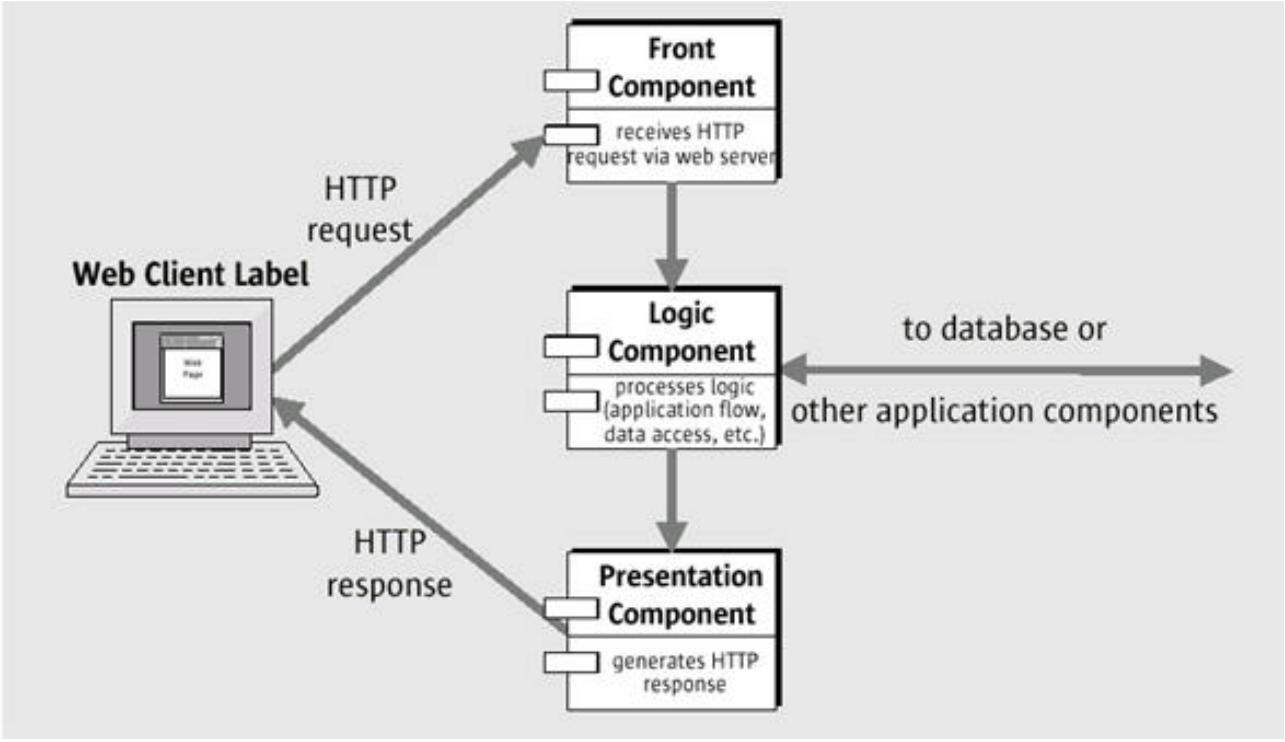


Figura 3.9: Front Component,Logic Component,Presentation Component

3.4.2 Interfaccia di Forte for Java

Il sistema operativo Windows deve il suo successo anche ad una interfaccia utente facile da usare, ed ha dettato sostanzialmente uno standard nel campo del design delle interfaccie. Un ambiente di sviluppo rapido nonostante si rivolga in genere ad un pubblico di utenti più esperti deve basare la propria interfaccia sui seguenti concetti:

1) Chiara ed intuitiva
2) Funzionale
3) Visione d'insieme del progetto

L'ambiente di sviluppo di Sun Microsystem ha un'interfaccia utente molto simile a quella del più famoso Visual Studio di Microsoft, permettendo dunque un facile utilizzo anche a coloro che avevano utilizzato in precedenza tale RAD. L'ambiente di sviluppo deve permettere in poche azioni l'esecuzione di comandi complessi per evitare inutile dispendio di tempo. Azioni tipiche come la ricompilazione di un progetto, oppure operazioni di copia incolla e altre operazioni che ricorrono di frequente devono essere facilmente realizzabili con pochi click del mouse. Per ultima, ma non per importanza, è la visione d'insieme del progetto, questa è fondamentale soprattutto se si lavora in team di sviluppo in quanto consente ad ogni progettista di verificare lo stato dei lavori del software.
L'ambiente di sviluppo Forte for Java è realizzato tenendo in considerazione anche queste caratteristiche.
Di seguito viene data una descrizione dell'aspetto del RAD al fine di capire meglio la struttura con la quale è realizzato. L'ambiente di sviluppo è costituito da diverse finestre:

- La barra degli strumenti
- Editor
- Explorer

La barra degli strumenti costituisce, ormai un classico di tutte le applicazioni, e consente di effettuare la compilazione, di eseguire le applicazioni di creare interfacce attraverso una *component palette* ecc. L'editor è dotato di funzioni di autocomposizione, supporta il copia-incolla, ed è dotato di highlights per il riconoscimento di parole chiave, oggetti ecc.
Ma uno degli strumenti più utili ed interessanti per la realizzazione di applicazioni web è senza dubbio l'explorer.

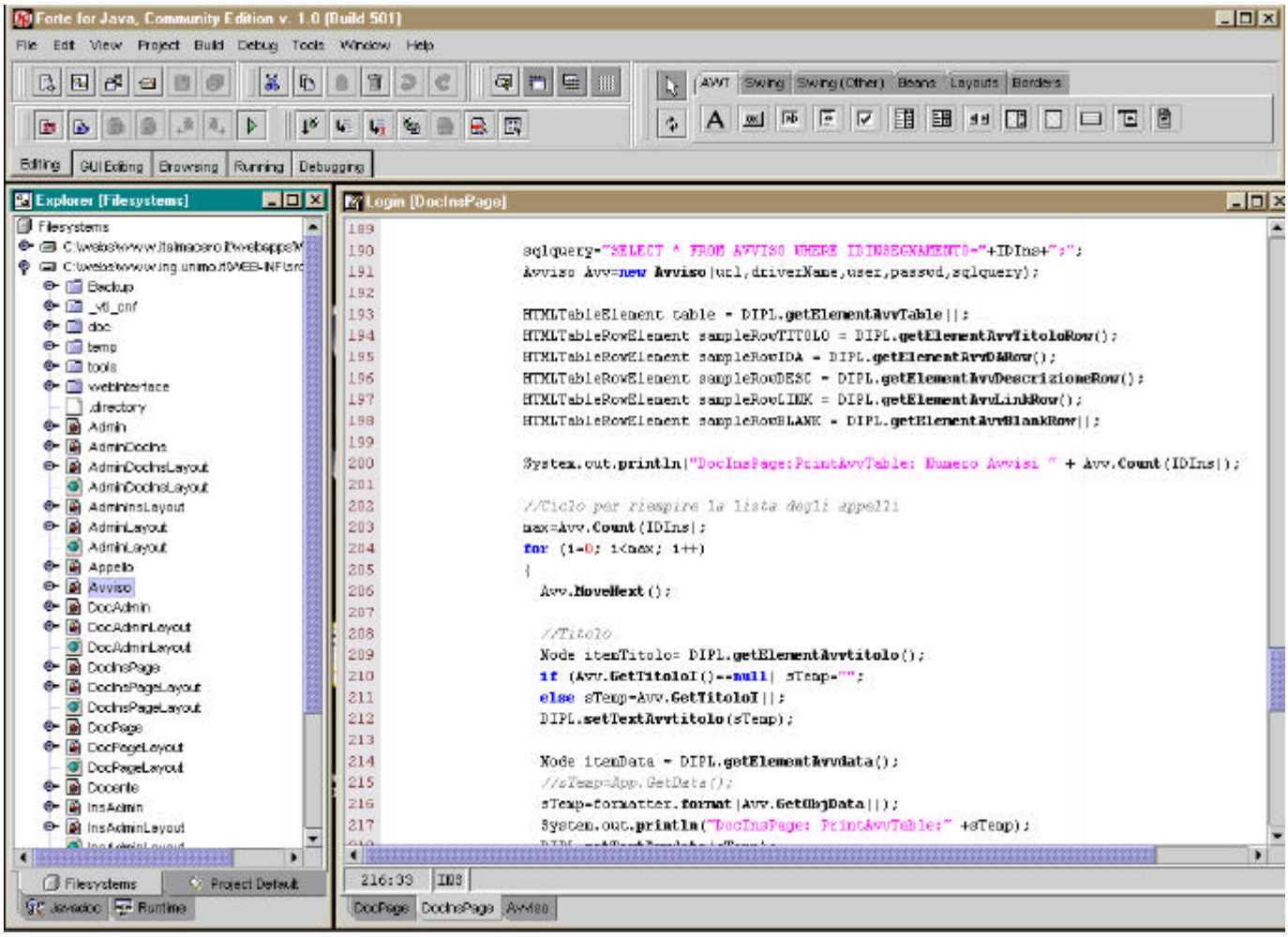


Figura 3.10: Forte for Java

Quest'ultimo consente infatti di montare porzioni di file system ed esplorarne il contenuto. Inoltre riconosce i file di classe java, e consente facilmente di visualizzare metodi e campi. Supporta una incredibile varietà di tipi di file permettendo di modificarli nell'editor. Naturalmente l'explorer consente di montare,creare ed utilizzare file system di tipo CVS, per la verifica della versione dei sorgenti, permettendo dunque di lavorare in team di sviluppo e su progetti molto estesi.

3.4.3 Realizzare applicazioni web con Forte

In primo luogo è necessario ricordare come deve essere realizzata la struttura delle gerarchia di directory nota con il nome di "*Standard Directory Layout*" per una applicazione web.

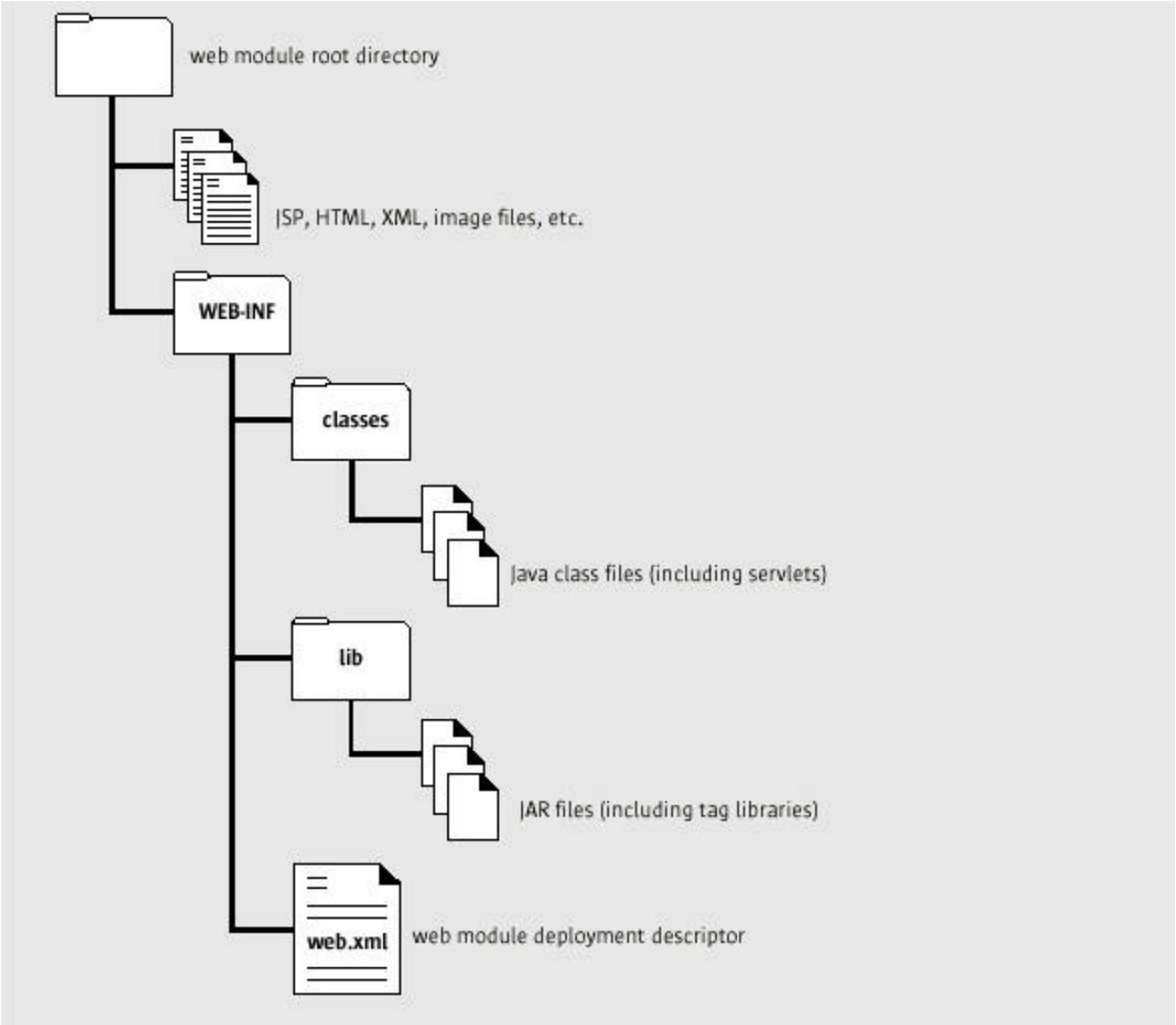


Figura 3.11: Standard Directory Layout

In Forte for Java la radice di questa gerarchia di directory prende il nome di "*web module root*", a partire da questa si sviluppano le altre directory fondamentali per la realizzazione della app licazione web, con la sola eccezione della directory "*src*" tipicamente preposta al contenimento dei sorgenti.

Come si può vedere dalla figura 3.X l'explorer di Forte riporta esattamente la struttura gerarchica di tale directory, consentendo una facile navigazione tra i file e una visione globale del progetto.

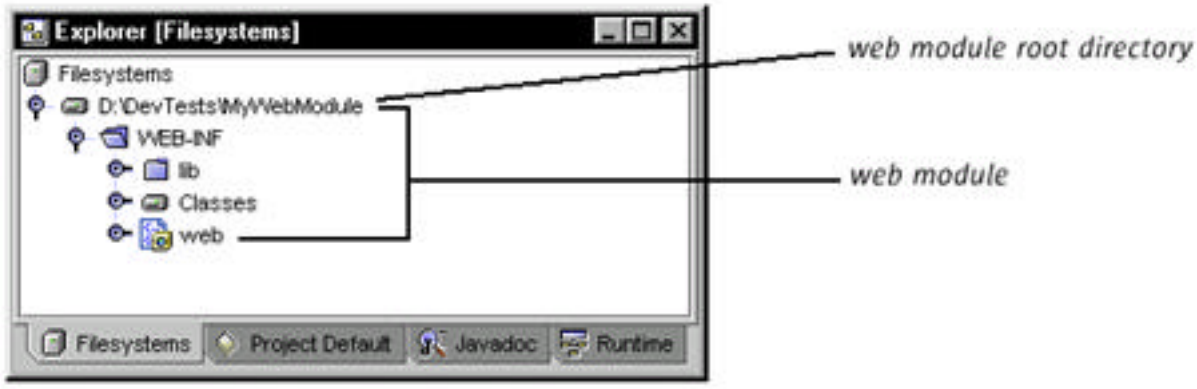


Figura 3.12: Explorer di Forte

Naturalmente è possibile aprire anche il file web.xml che corrisponde al "*web container*" e visualizzarne la struttura DOM per una immediata modifica dei parametri di configurazione della applicazione web.

Per testare l'applicazione realizzata è sufficiente scegliere quale è la pagina da visualizzare per prima e fare click sul pulsante "*Execute*" . Così facendo l'application server integrato di Forte eseguirà la servlet o la JSP scelta.

3.5 Cenni su XML/XSL

Una tra le tecnologie più importanti e che si sta maggiormente affermando per la realizzazione di siti web dinamici efficienti e di facile manutenzione, è quella che unisce il linguaggio XML e i fogli di stile XSL (eXtensible Stilesheet Language). Questa con buona probabilità sarà una delle tecnologie che nel prossimo futuro saranno maggiormente utilizzate, anche perché, senza volere togliere nulla alla bontà della tecnologia, le maggiori software house mondiali stanno spingendo notevolmente in tale senso. Basti pensare che i più recenti browser web come Microsoft Internet Explorer 5 (e versioni successive) e Netscape Navigator 6 supportano questa tecnologia in modo nativo.

3.5.1 Introduzione a XML

XML è l'acronimo di eXtensible Markup Language è un sottoinsieme del più completo SGML. XML è stato realizzato e standardizzato dal Worl Wide Web Consortium (W3C), nato come linguaggio di descrizione è diventato a tutti gli effetti uno standard per lo scambio di dati soprattutto su Internet, ma non solo. La potenza dell'XML sostanzialmente risiede nella sua semplicità è possibile infatti descrivere in modo molto chiaro e in un formato che non lascia interpretazioni personali dei dati anche una struttura complessa, di fatto è un linguaggio semistrutturato. I dati sono interpretabili in modo univoco e questo lo rende ideale per lo scambio di qualsiasi informazione indipendentemente dall'architettura hardware o dal sistema operativo dei computer utilizzati.
Il linguaggio XML si basa su file di testo contenenti tag definiti nel file di testo medesimo. Come per HTML con l'unica differenza che in quest'ultimo i tag sono predefiniti dalla sintassi del linguaggio stesso. In effetti sotto opportune condizioni l'HTML si può vedere come un sottoinsieme del XML e in tale caso prende il nome di XHTML. Addirittura anche le Java Server Pages sono dei documenti XML. Ma dove il linguaggio esprime veramente il massimo è nella rappresentazione di dati, ad esempio nell'esprimere il risultato di una query. In altre parole fornisce una rappresentazione dei dai in un formato facilmente leggibile. Basti pensare che in un qualsiasi sito web dinamico l'interazione con una base di dati è fondamentale, e un modo per rappresentare il risultato di una interrogazione è appunto XML. Ovviamente questo da solo non è sufficiente per

definire il layout di una pagina, per tale motivo si stanno studiano soluzioni per rendere il estendere il formato XML con capacità grafiche.

3.5.2 Introduzione a XSL

La necessità di fornire una rappresentazione grafica di una pagina ha fatto sì che si venisse studiato il modo di applicare un foglio di stile ad un file XML, nasce dunque l'esigenza di definire un nuovo standard che prendere il nome di XSL (eXtensible Stylesheet Language). XSL a sua volta può essere visto come una estensione di XML. Il linguaggio per realizzare fogli di stile è sostanzialmente composto da due parti:

- XSL Transformation
- XSL Formatting Object

Il primo è definisce le regole di trasformazione da un documento XML ad un altro. Il secondo sostanzialmente costituisce una potente estensione a ciò che veniva realizzato attraverso i fogli di stile CSS per il linguaggio HTML. La parte di formattazione di oggetti di XSL ha un vocabolario molto più esteso rispetto a CSS e per tale motivo è possibile personalizzare ogni documenti fino ad un livello notevole.

Tuttavia la parte più utilizzata ed importante dell'XSL è proprio il *transformer*, il cui uso attuale più diffuso è quello di realizzare a partire da un documento XML un documento XHTML, che sia interpretabile da qualsiasi browser. Un altro vantaggio che deriva utilizzando il trasformer è la possibilità di cambiare il tipo di linguaggio target, ad esempio è possibile passare da HTML a WML.

3.5.3 Cocoon

Cocoon, come Tomcat, è realizzato dall'Apache Software Fondation, ma fa parte di un progetto differente che prende il nome di XML Apache. In questa sezione del progetto, vengono studiate tutti i tipi di tecnologie che coinvolgono il linguaggio XML, e che possono avere qualche risvolto nella realizzazione di applicazioni web.

Cocoon è uno strumento di Web Publishing di recente sviluppo realizzato interamente in Java che permette la realizzazione di pagine dinamiche a partire da file XML e file XSL. Il vantaggio che

deriva utilizzando l’accoppiamento XML e XSL è considerevole avendo a disposizione uno strumento di publishing come Cocoon . Basta pensare alla semplice architettura proposta. Si supponga di realizzare un progetto che effettui un qualsiasi accesso ad un database e rappresenti il risultato delle query in formato XML, a questo punto basta avere realizzato un file XSL che contenga la presentazione della pagina, e Cocoon pubblica il risultato della trasformazione in un file XHTML. Il vantaggio è notevole in quanto si è realizzato completamente la separazione tra la presentazione cioè l’aspetto grafico della pagina, e il suo contenuto informativo. Naturalmente questa è solamente una delle funzioni possibili con questa libreria.

3.6 Confronto tra tecnologie

Può risultare utile paragonare le tecnologie analizzate in questo capitolo, in particolare si vogliono confrontare la tecnica XMLC e il linguaggio XML/XSL.

Entrambe le tecnologie si basano sull'XML, ma il risultato finale è la rappresentazione di un documento HTML.

XMLC è una libreria che trasforma un file HTML in una classe Java, e consente di utilizzare la classe come template della pagina da realizzare. Inoltre è possibile includere nella pagina alcuni tag che consentono di individuare aree di testo importanti che possono essere soggette a modifiche. Il vantaggio che deriva dall'utilizzo di questa tecnologia è dato dalla possibilità di delegare a grafici la realizzazione della pagina web, e poi realizzare tutta la logica della applicazione attraverso servlet. Quindi la presentazione è di facile realizzazione.

XML/XSL fornisce invece una maggiore separazione tra quello che sono i dati che possono venire estrapolati attraverso l'uso di servlet, e la presentazione che è realizzata attraverso un foglio di stile. Allo stato attuale dei lavori, l'unico possibile punto debole di questo tipo di tecnica è senz'altro il foglio di stile. Infatti sono solamente da poco state definite le specifiche dei foglio di stile in modo standard, e per tale motivo attualmente non esiste alcun software in grado di agevolare la realizzazione di documenti XSL. Questo significa che non è ancora possibile delegare a terze parti l'aspetto grafico di una pagina web realizzata con il linguaggio XML/XSL, c'è da dire a ragion del vero che un documento XSL nella sua interpretazione più semplice è molto simile a un documento HTML. Nonostante questa mancanza, la realizzazione di siti web basati su tecnologia XML/XSL porta ad avere pagine dinamiche facilmente mantenibili, a tutto vantaggio quindi della facilità di aggiornamento della veste grafica e dei contenuti del sito.

Capitolo 4
Scelte Tecnologiche

In questo capitolo si mettono in evidenza quali sono state le scelte tecnologiche per la realizzazione del prototipo della applicazione. Inoltre si forniscono alcune informazioni sulle fasi implementative del progetto.

4.1 Tecnologie Utilizzate

Le tecnologie utilizzate per la realizzazione della applicazione web coinvolgono l'uso di servlet per la creazione pagine web dinamiche abbinate all'uso del compilatore XML di Lutris Tecnology. La scelta di utilizzare XMLC è stata presa in funzione della faciltà d'uso del compilatore stesso e dalle garanzie di mantenibilità futura del contenuto grafico del sito. Inoltre l'utilizzo di XMLC ha reso possibile lo sviluppo della applicazione in un tempo abbastanza limitato. Come application server si è fatto uso di Tomcat e come ambiente di sviluppo di Forte for Java.

4.2 Approccio Modulare

Come già visto nel capitolo precedente il progetto di una applicazione web può essere orientato su un approccio del tipo web centric oppure uno più modulare.
Naturalmente per un progetto delle dimensioni di quello del portale web di facoltà, non era possibile utilizzare l'approccio web-centric, per tale motivo si è scelto l'approccio di tipo modulare. La tecnica utilizzata si basa principalmente sulla realizzazione di classi specifiche per l'accesso ai dati così come è stato definito nella fase analisi, invece per ciò che riguarda la presentazione di una pagina dinamica si fa uso di servlet e XMLC, così come l'acquisizione di informazioni che è a sua volta realizzato attraverso l'uso di form HTML oppure servlet e XMLC.

In figura è mostrato l'approccio utilizzato attraverso un diagramma:

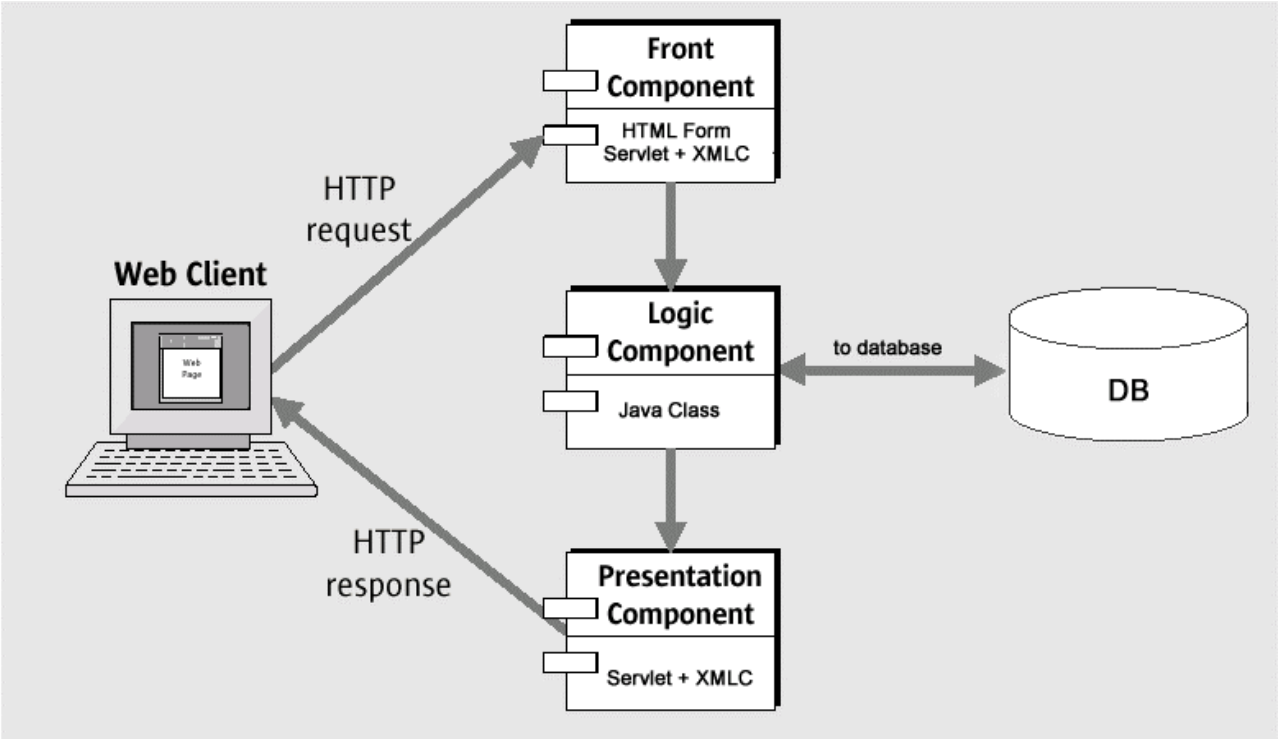


Figura 4.1: Approccio utilizzato per la realizzazione della applicazione

La struttura realizzata fornisce buone prestazioni ed ha il vantaggio di essere facilmente comprensibile, inoltre, qualche accorgimento nell'uso di XMLC, facilita di gran lunga le operazioni di manutenzione del sito.
Nel caso in cui si renda necessario è possibile estendere la parte di presentazione e di acquisizione di informazioni attraverso l'uso di Java Server Page, quindi la struttura proposta rimane aperta per ogni modifica. Inoltre separare la logica dell'applicazione fa sì che una modifica su una delle tabelle contenute nel database, rispecchi solamente una modifica nella classe che si occupa dell'accesso ai dati per quella tabella.

Per la realizzazione della parte di presentazione si è fatto uso solamente delle caratteristiche di XMLC. Dapprima si sono realizzati tutti i template dei documenti HTML, inserendo dove opportuno alcuni tag identificativi, poi con il compilatore si sono convertiti in classi Java, in modo tale da poter essere utilizzate all'interno di servlet. Una modifica ai template HTML richiede una riconversione del file in classe Java, a meno che questo meccanismo non venga realizzato automaticamente attraverso codice oppure tramite uno script.

4.3 Gestione delle sessioni

Di seguito si presentano le tecniche più diffuse per la gestione delle sessioni in una applicazione e poi si mostra la scelta effettuata per il portale di facoltà.

Durante la realizzazione di una sito web dinamico di una certa complessità, si rende prima o poi necessario tenere traccia delle azioni compiute dall'utente oppure scambiare dati tra una pagina e l'altra. Questa operazione può non essere banale perché come già accennato nel capitolo precedente il protocollo HTTP è stateless, in questo protocollo infatti un client opera una connessione richiedendo alcune risorse o informazioni, il server risponde mettendo a disposizione del client le risorse richieste se disponibili, oppure invia una messaggio di errore secondo il protocollo HTTP. Una volta chiusa la connessione il server non ricorda alcuna informazione relativa al computer client che ha effettuato la connessione. In questo modo il server considera ogni richiesta da parte di un client come una nuova richiesta, senza alcuna relazione con la precedente. Un protocollo è statefull se la risposta ad una precisa richiesta da parte del client può essere programmata e dipende non solo dalla richiesta corrente ma anche da quelle precedenti. L'importanza di un protocollo statefull è evidente quando si ha a che fare con applicazioni complesse che effettuano richieste e risposte multiple. Si consideri ad esempio una applicazione di banking online, quando si richiede una pagina contenente il bilancio dei propri conti, il server deve essere in grado di verificare l'identità del possessore dei conti attraverso uno scambio di credenziali (ad esempio username e password) con l'applicazione di banking. Tuttavia a causa del protocollo stateless ad ogni singola operazione sarebbe necessario rinviare le proprie credenziali. Ovviamente ciò è impensabile per lunghe operazioni bancarie. Un altro tipico esempio è quello del carrello elettronico dove è necessario memorizzare gli acquisti effettuati dall'utente.

Al fine di risolvere il problema dovrebbe essere possibile utilizzare i due seguenti metodi:

- Sessioni – Il server dovrebbe essere in grado di identificare quella serie di richieste che avvengono da un singolo client. E' indispensabile realizzare questa caratteristica ed è possibile farlo associando ad una specifica richiesta una sessione di lavoro.
- Stato – Il server dovrebbe essere in grado di memorizzare le informazioni richieste in precedenza. L'applicazione dovrebbe essere in grado di associare ad ogni stato una particolare momento di una sessione.

Tuttavia le connessioni con il protocollo HTTP vengono chiuse al termine di ogni richiesta, di conseguenza per il server è impossibile usare il concetto di connessione per definire una sessione.
HTTP è un protocollo stateless, per ciò che riguarda richieste e risposte e questo fondamentalmente si può vedere come se avvenissero in continuazione transazioni isolate l'una dall'altra tra il client e il server. Naturalmente questa situazione è perfetta per ciò che riguarda la navigazione di un sito attraverso pagine statiche, in quanto normalmente si tratta di semplici operazioni di download, per il server non è importante che queste richieste avvengano da un singolo client o da clienti distinti. Il discorso cambia radicalmente quando si ha a che fare con applicazioni web, dove invece deve essere possibile effettuare transazioni attraverso richieste multiple e risposte multiple tra client differenti. Il problema si può anche riassumere dicendo che il server non riesce a determinare se le richieste arrivano tutte dallo stesso client, d'altro canto però il client non può iniziare una comunicazione con il server web per effettuare una transazione.
Oltre al discorso di tenere traccia degli utenti, basato sulla concetto di sessione, il server dovrebbe essere in grado di ricordare anche ogni informazione associata alla specifica sessione. Questo è ciò che i programmatori di applicazioni web devono fare ogni volta si renda necessario memorizzare i alcuni dati di una specifica sessione.

Nel tempo sono state sviluppate diverse strategie per risolvere il problema della sessione e dello stato di una sessione. Le Java Servlet API forniscono alcune tecniche per tracciare a mantenere lo stato di una sessione. Grazie all'uso di alcune classi e di oggetti il server è in grado di riconoscere quali tra le richieste avvengono da un singolo client e può memorizzare lo stato della sessione.

Ci sono sostanzialmente quattro approcci possibili per mantenere la traccia di una sessione:

- URL rewriting
- Cookies
- Hidden Form Field
- Sessioni utilizzando Secure Sockets Layer (SSL)

Comunque tutti e quattro questi approcci si basano sul fatto che deve avvenire in qualche modo uno scambio di un *"token"* tra il client e il server. Consideriamo ad esempio un client C ed un server S. Quando C invia una richiesta a S per la prima volta, S fornisce a C un *token* univoco. Ogniqualvolta che C visita di nuovo S, ritrasmette il *token* assieme alla richiesta. Con questo semplice metodo il server è in grado di riconoscere il client grazie al suo token.

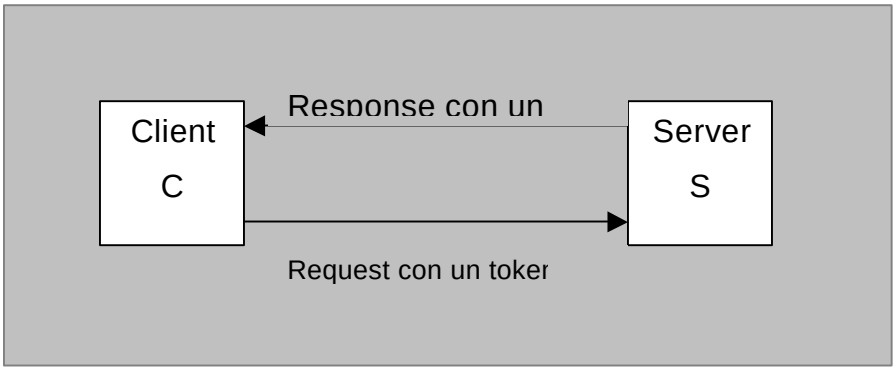


Figura 4.2: Scambio di un token

Questa semplice tecnica per tenere traccia delle sessioni può essere realizzata in modi diversi, basati sulla modalità di scambio del token e sulla sua tipologia. Di seguito si analizzano brevemente le quattro tecniche per mantenere traccia delle sessioni

Nella applicazione web sviluppata per il portale di facoltà, si fa uso sia del concetto di sessione, che di quello di Hidden Form Field e in parte anche di URL Rewriting. L'uso di Hidden Form Field risulta particolarmente comodo laddove è necessario specificare un gran numero di dati, ma non si vuole che l'utente interagisca con questi. Inoltre grazie ad XMLC è possibile cambiare questa politica e consentire ad un utente l'interazione con i dati nascosti ad esempio cambiando il valore di un campo dal tipo "Hidden" al tipo "Text" senza alterare alcuna parte del codice.

4.3.1 URL Rewriting

In questo tipo di tecnica il token è incorporato in ogni URL. Infatti in tutte le pagine generate dinamicamente il server incorpora un parametro extra ad esempio estratto da una query. Quando il client, effettua la sua richiesta utilizzando l'URL, il token viene ritrasmesso al server. Questo approccio viene chiamato URL rewriting proprio perché ad ogni pagina dinamica viene riscritta insieme all'URL anche una stringa che identifica la sessione.

4.3.2 Hidden Form Fields

Questo approccio è simile all'URL rewriting. Tuttavia invece di riscrivere ogni URL, dove risulta necessario alcuni variabili significative per l'applicazione vengono nascosti nei campi di tipo "hidden" di ogni form HTML. Quando il client effettua una richiesta invia inconsapevolmente

anche i valori contenuti in questi campi. Il server può utilizzare i valori delle variabili per tenere traccia delle sessioni.

4.3.3 Cookies

Il termine cookie la cui traduzione letterale ha il significato di biscotto, è una delle forme utilizzate tra client e server per scambiare informazioni. Inventato da Netscape, si basa su un concetto di funzionamento differente rispetto all'URL rewriting e agli Hidden form fields. I cookie possono essere scambiati tra gli header delle operazioni di request e response.

4.3.4 Secure Socket Layer

SSL è una tecnologia di criptazione che gira sul protocollo TCP/IP and a livello della applicazione sull'http. Sostanzialmente questa tecnologia è quella utilizzata nel protocollo HTTPS. SSL permettette ai server che dispongono di questo protocollo di autentificare i client e di mantenere una connessione criptata tra i due. Nel momento di creazione della connessione viene generato un *"session id"* univoco che viene utilizzato per effettuare la decriptazione dei messaggi.

Nel caso di sviluppo di applicazioni tramite l'uso di servlet, è il web container che si occupa di gestire le sessioni, nonostante ciò è possibile utilizzare congiuntamente anche qualche altro criterio. Le API delle servlet forniscono una interfaccia che è di importanza fondamentale per manipolare correttamente le sessioni.

4.3.5 Sicurezza dell'applicazione

La necessità di garantire la sicurezza delle informazioni pubblicate ha fatto sì che in tutte le pagine dinamiche dove è possibile interagire con i dati del personale o qualsiasi altra informazioni che risulti essere in qual modo sensibile, venga tutelata utilizzando i criteri di sicurezza messa a disposizione dalla tecnologia scelta per lo sviluppo della applicazione.
In questi termini è possibile operare in due modi:

- Sul sistema utilizzando protocolli sicuri ad esempio HTTPS.
- A livello di programmazione utilizzando il concetto di sessione.

Allo stato attuale dei lavori, è stato implementato solamente il secondo modo, ma una volta che il progetto verrà rilasciato sicuramente la parte di amministrazione del sito utilizzerà un protocollo di

comunicazione sicura, in modo tale da rendere molto difficile l'intercettazione di dati sensibili lungo la rete.

4.4 Organizzazione dei sorgenti

Affinché il progetto risulti il più possibile chiaro e intuitivo e conseguentemente di facile manutenzione si è pensato di organizzare i sorgenti in modo tale che i template HTML e le servlet avessero un nome che ne ricordasse la funzione.

Ad esempio tutti i template HTML hanno come nome xxxLayout.java in modo tale da individuarli a colpo d'occhio. Inoltre tutte le pagine dinamiche a sola lettura, cioè quelle che non consentono interazione hanno come nome xxxPage.java. Infine le servlet per pagine di amministrazione contengono la stringa xxxAdmin.java. Combinando queste regole ed eventualmente usando i package di java si riesce ad ottenere una ottima organizzazione dei sorgenti. Ad esempio il file "DocAdminLayout.class" corrisponde al file di classe ottenuto tramite XMLC come template per la parte di amministrazione dei docenti.

4.5 Schema relazionale

Come è già stato detto l'applicazione web realizzata fa largo uso di accesso ad un database, attraverso il RDBMS Interbase 6 di Borland. Una delle fasi più importanti nella realizzazione di qualsiasi applicazione di carattere gestionale è lo realizzazione di un corretto schema relazionale.
La realizzazione dello schema E/R riguardante la realtà da modellare nell'ambito della facoltà di Ingegneria Informatica è stato realizzato in modo veramente esauriente in una tesi precedenti. Inoltre lo schema relazionale è stato prodotto a partire da quello schema E/R nel progetto di facoltà. Tuttavia per completezza di contenuti e per inquadrare la realtà del progetto realizzato è necessario riportare una parte di quello schema.

Studente(<u>userName</u>, matricola,nome,cognome)

FK userName REFERENCES Utente

Docente(userName, idDocente,nome,cognome,cf, ricevimento, ricerca,
affiliazioni,cooperazioni,foto,…)

FK userName REFERENCES Utente

AK idDocente

DocenteEsterno(userName)

FK userName REFERENCES Docente

FK userName REFERENCES Collaboratore

DocenteInterno(userName , sername o)

FK userName REFERENCES Dipendente

FK userName REFERENCES Docente

Professore(userName)

FK userName REFERENCES DocenteInterno

Ricercatore(userName)

FK userName REFERENCES DocenteInterno

Prof_II_Fascia(userName)

FK sername REFERENCES Professore

Prof_I_Fascia(userName)

FK sername REFERENCES Professore

Insegnamento(idInsegnamento, idDocente, nome, descrizione, programma, progprecedente)

Appello(idInsegnamento, data, ora, aula,descrizione)

FK idInsegnamento REFERENCES Insegnamento

Avviso(idInsegnamento, data, autore,titoloi,titoloe,descrizionei,descrizionen,linkfile)

FK idInsegnamento REFERENCES Insegnamento

Materiale(idInsegnamento, idMateriale, annoacc,descrizione,linkfile)

Si riportano per completezza anche le istruzioni DDL necessarie per ricreare le tabelle relative allo schema relazionale. E' stato possibile reperire facilmente queste informazioni attraverso la console di Interbase. Questo tool che funziona sotto Windows permette di eseguire semplici interrogazioni al database, gestire backup e visualizzare i metadata che costituiscono le tabelle.

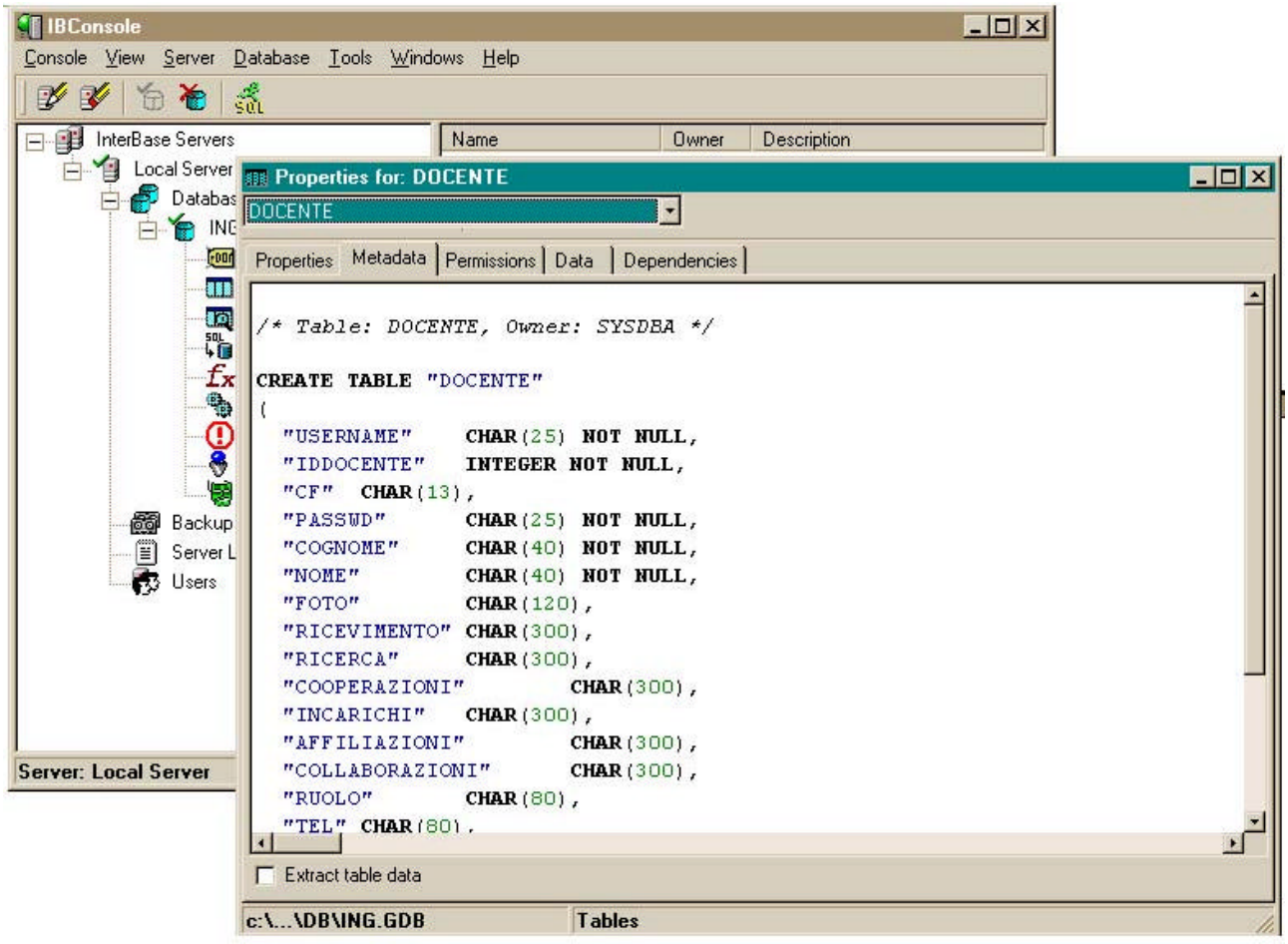


Figura 4.3: Estrazione di metadata dalla IBConsole

Istruzioni DDL per la creazione delle tabelle:

?? Tabella dei docenti

```
CREATE TABLE "DOCENTE"
```

```
(
  "USERNAME"      CHAR(25) NOT NULL,
  "IDDOCENTE"     INTEGER NOT NULL,
  "CF"      CHAR(13),
  "PASSWD"  CHAR(25) NOT NULL,
  "COGNOME" CHAR(40) NOT NULL,
  "NOME"    CHAR(40) NOT NULL,
  "FOTO"    CHAR(120),
  "RICEVIMENTO"   CHAR(300),
  "RICERCA" CHAR(300),
  "COOPERAZIONI"  CHAR(300),
  "INCARICHI"     CHAR(300),
  "AFFILIAZIONI"  CHAR(300),
  "COLLABORAZIONI"      CHAR(300),
  "RUOLO"   CHAR(80),
  "TEL"     CHAR(80),
  "FAX"     CHAR(80),
  "EMAIL"   CHAR(80),
  "UFFICIO" CHAR(300),
  "LINK"    CHAR(300),
CONSTRAINT "DOCENTE_PK" PRIMARY KEY ("USERNAME")
);
```

Al fine di evitare di complicare eccessivamente la struttura del database si è preferito realizzare un collasso verso l'alto della gerarchia dei docenti. Questa modifica implica l'inserimento di un nuovo attributo nella relativa tabella chiamato ruolo. L'attributo può assumere i valori di: "DocenteInterno","DocenteEsterno","Ricercatore","Professore", ecc. Inoltre è stata inserita la chiave alternata "idDocente" che risulta di estrema utilità nelle pagine dinamiche del sito per identificare univocamente un docente senza svelare il suo username.

?? Tabella degli insegnamenti

```
CREATE TABLE "INSEGNAMENTO"
(
  "IDDOCENTE"     INTEGER NOT NULL,
  "IDINSEGNAMENTO"      INTEGER NOT NULL,
  "NOME"    CHAR(80) NOT NULL,
  "DESCRIZIONE"   CHAR(500) NOT NULL,
  "PROGRAMMA"     CHAR(1000),
  "PROGPRECEDENTE"      CHAR(1000),
CONSTRAINT "INSEGNAMENTO_PK" PRIMARY KEY ("IDINSEGNAMENTO")
);
```

?? Tabella degli appelli

```
CREATE TABLE "APPELLO"
(
 "DATA"                    TIMESTAMP NOT NULL,
 "IDINSEGNAMENTO"          INTEGER NOT NULL,
 "ORA"                     CHAR(10) NOT NULL,
 "AULA"                    CHAR(80) NOT NULL,
 "DESCRIZIONE"             CHAR(100),
CONSTRAINT "APPELLO_PK" PRIMARY KEY ("IDINSEGNAMENTO", "DATA")
);
```

?? Tabella degli avvisi

```
CREATE TABLE "AVVISO"
(
  "IDAVVISO"      INTEGER NOT NULL,
  "IDINSEGNAMENTO"      INTEGER NOT NULL,
  "DATA"    TIMESTAMP NOT NULL,
  "AUTORE"  CHAR(80) NOT NULL,
  "TITOLOI" CHAR(300) NOT NULL,
  "TITOLOE" CHAR(300),
  "DESCRIZIONEEN" CHAR(300),
  "LINKFILE"      CHAR(120),
  "DESCRIZIONEI"  CHAR(300) NOT NULL,
 PRIMARY KEY ("IDAVVISO")
);
```

?? Tabella del materiale didattico

```
CREATE TABLE "MATERIALE"
(
  "IDMATERIALE"   INTEGER NOT NULL,
  "IDINSEGNAMENTO"      INTEGER NOT NULL,
  "ANNOACC" CHAR(20) NOT NULL,
  "DESCRIZIONE"   CHAR(300),
  "LINKFILE"      CHAR(200),
CONSTRAINT "MATERIALE_PK" PRIMARY KEY ("IDMATERIALE")
);
```

Appendice A

Fino dal rilascio della prima versione nell'autunno del 1999, il cambiamento più significativo in Java, forse è stata l'introduzione della piattaforma **Java 2 Enterprise Edition** (J2EE), particolarmente pensata per l'uso server-side. J2EE rappresenta il tentativo da parte della Sun Microsystem di rendere Java, non solo un linguaggio di sviluppo, ma una più importante piattaforma adatta allo sviluppo di applicazione in ambito Enterprise. Il primo obiettivo nel realizzare J2EE è stato quello di incapsulare molte dei componenti di livello enterprise, come ad esempio le connection pooling e le transazioni in ciò che si può definire una architettura **contatiner-based**. Ciò che rimane da realizzare allo sviluppatore, almeno in linea teorica, è l a business logic della sua applicazione. In figura è possibile vedere un diagramma dei componenti fondamentali che costituiscono la piattaforma di J2EE.

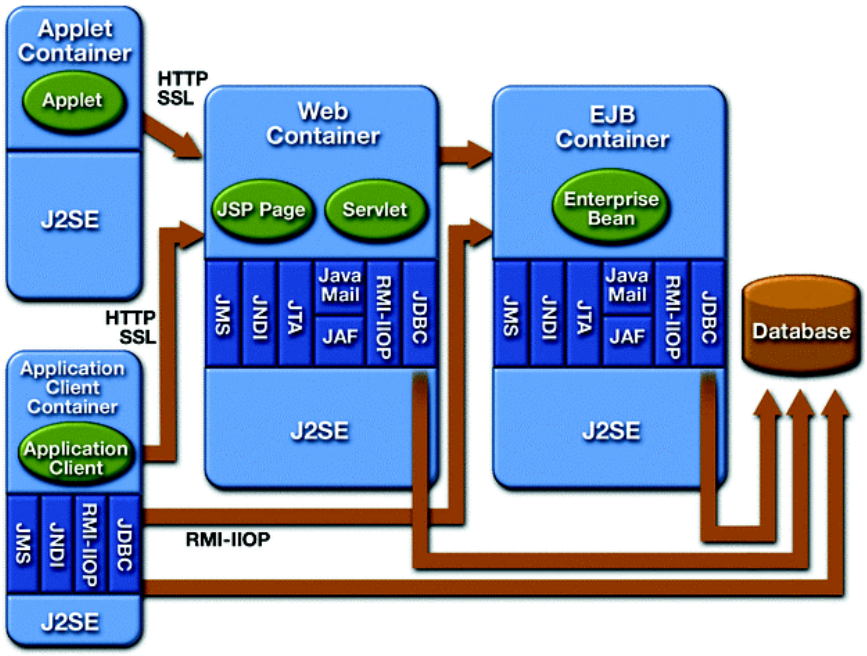


Figura A.1: **J2EE Architecture**

Questa versione di Java è pensata per lo sviluppo di applicazioni lato server, e prime fra tutte, le applicazioni web based di una certa importanza. J2EE risponde alle caratteristiche richieste da un programmatore di applicazioni enterprise che sono:

?? Produttività nella programmazione – Avere a disposizioni nuove oggetti e classi raggruppati e definiti in modo efficiente, aumenta la velocità con cui viene realizzata una nuova applicazione.

?? Affidabilità e Disponibilità – Nell'epoca di Internet importantissima è l'affidabilità di un sistema, ciò che è definito **downtime** (ciò l'intervallo temporale in cui un server o una applicazione non sono in grado di fornire il servizio a cui erano dedicati) deve essere minimo o inesistente.

?? Sicurezza – Lo sviluppo di applicazioni web-based aumenta notevolmente il numero di utenti che possono interagire con il sistema, per questo è di prima importanza realizzare applicazioni sicure. Ovviamente più una tecnologia diventa avanzata, le applicazioni più sofisticate, implementare un buon livello di sicurezza diventa sempre più difficile.

?? Scalabilità – La capacità di una applicazione di crescere e di soddisfare le nuove richieste di operazioni e utenti è molto importante, specialmente quando si pensa alle potenziale migliaia di utenti che può avere un sito web. La scalabilità in questo senso va intesa anche come la capacità di sfruttare al meglio le risorse del sistema, non solo come la capacità di soddisfare un numero maggiore di richieste dei client.

?? Integrabilità – Spesso durante lo sviluppo di applicazioni di una certa importanza, si ha a che fare con dati esistenti e con esigenze consolidate. L'applicazione da realizzare dunque, deve essere in grado di integrarsi con il sistema esistente e di estenderne le funzionalità. La capacità di combinare tecnologie vecchie e nuove è una dei concetti fondamentali da tenere in considerazione.

System Architecture

Quando si parla di sviluppo di applicazioni enterprise, risulta necessario introdurre il concetto di architettura **n-tier** (o multi-tier). I tipici sistemi client-server sono basati su una architetturara 2-tier, dove è evidente la separazione tra i dati e la presentation e la business logic. Questo tipo di applicazioni che girano principalmente sul client sono spesso definite data-driven, il secondo strato della applicazione in genere è costituito dal database server.

2-Tier Architecture

In una applicazione tipica 2-tier, il carico di lavoro pesa interamente sul PC client, mentre il server semplicemente agisce da "controllore del traffico" tra l'applicazione e i dati. Come risultato si può avere che le prestazioni della applicazione sia limitate a causa delle scarse risorse del PC. Quando l'intera applicazione viene eseguita sul PC, questa può effettuare richieste multiple per dati, prima di presentare una risposta all'utente, e a causa di queste richi este, il traffico di rete tende ad aumentare notevolmente.

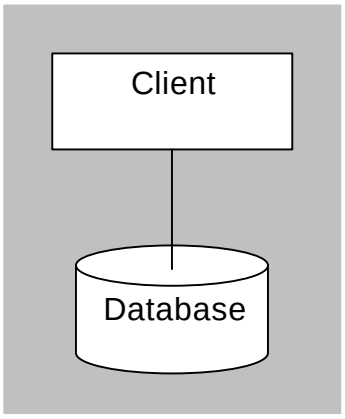


Figura A.2: 2 Tier Architecture

Un altro grave problema delle applicazioni basate sull'approccio a due strati è la manutenzione. Infatti un piccolo cambiamento nella applicazione (ad esempio dovuto alla correzione di un bug) necessita la reinstallazione su tutti i client. Fortunatamente alcuni tool di sviluppo permettono almeno in parte di automatizzare questo processo. Nel casopiù sfortunato può accadere che alcuni utenti ignorino che la disponibilità di una nuova versione della applicazione e continuino ad usare quella meno recente con la grave conseguenza che esistano più versioni della applicazione che continuano a girare nello stesso ambito aziendale.

3-Tier Architecture

Per risolvere alcuni dei problemi introdotti con l'approccio client-server, si è introdotta l'architettura 3-Tier. Si può immaginare di dividere l'applicazione in tre strati logici differenti ognuno con una interfaccia ben definita. Il primo strato viene definito **presentation layer** ed in genere consiste in un qualsiasi tipo di interfaccia grafica utente (GUI). Lo stato di mezzo viene detto **business layer**, realizza l'applicazione o la business logic. Infine il terzo strato, **data layer**, contiene i dati necessari per l'applicazione.

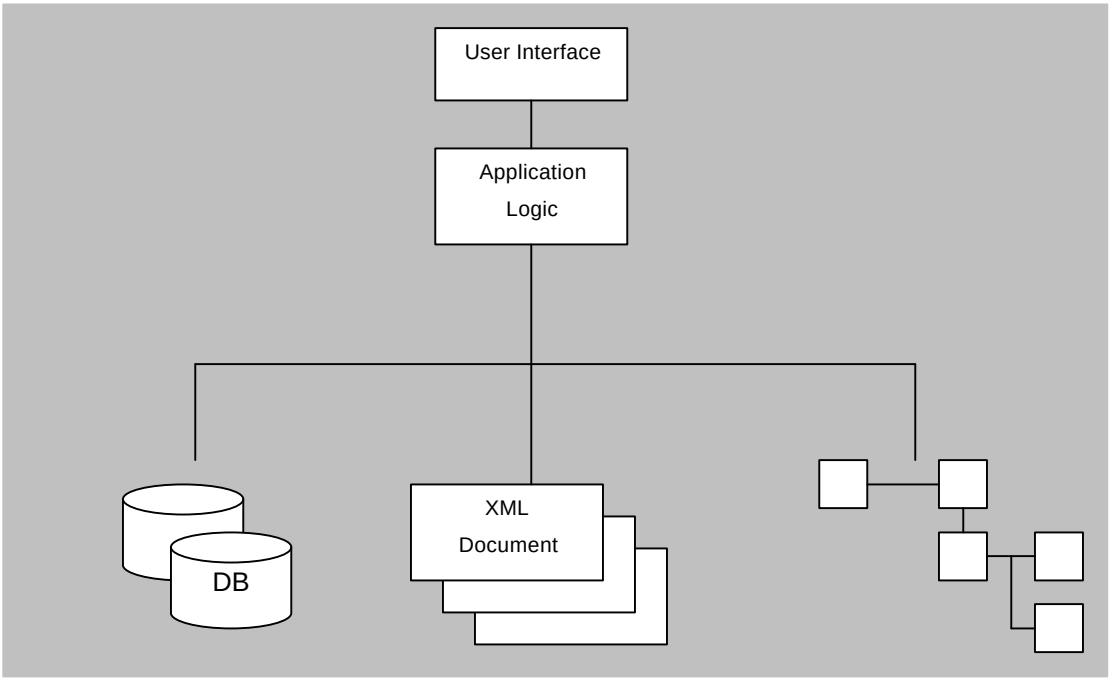


Figura A.3: 3-Tier Architecture

Lo strato centrale è in genere costituito dal codice che l'utente richiama attraverso l'interfaccia grafica, al fine di avere i dati desiderati. Il presentatio layer riceve i dati e li formatta in modo adeguato per la loro visualizzazione. Ovviamente la separazione tra la logica della applicazione e l'interfaccia utente porta incredibili vantaggi e una ottima flessibilità al progetto della applicazione. In linea teorica è possibile realizzare più di una interfaccia grafica senza cambiare la business logic della applicazione, in quanto non c'è interferenza tra i due strati, ma solo comunicazione.

Il terzo e ultimo strato contiene i dati, questi possono essere forniti attraverso una qualsiasi sorgente di informazione, come ad esempio un RDBMS, oppure un insieme di documenti XML o ancora un server LDAP.

n-Tier Architecture

Un sistema multi strato non è definibile in modo preciso, tuttavia è possibile dare una linea generica che normalmente è seguita da questo tipo di applicazioni.

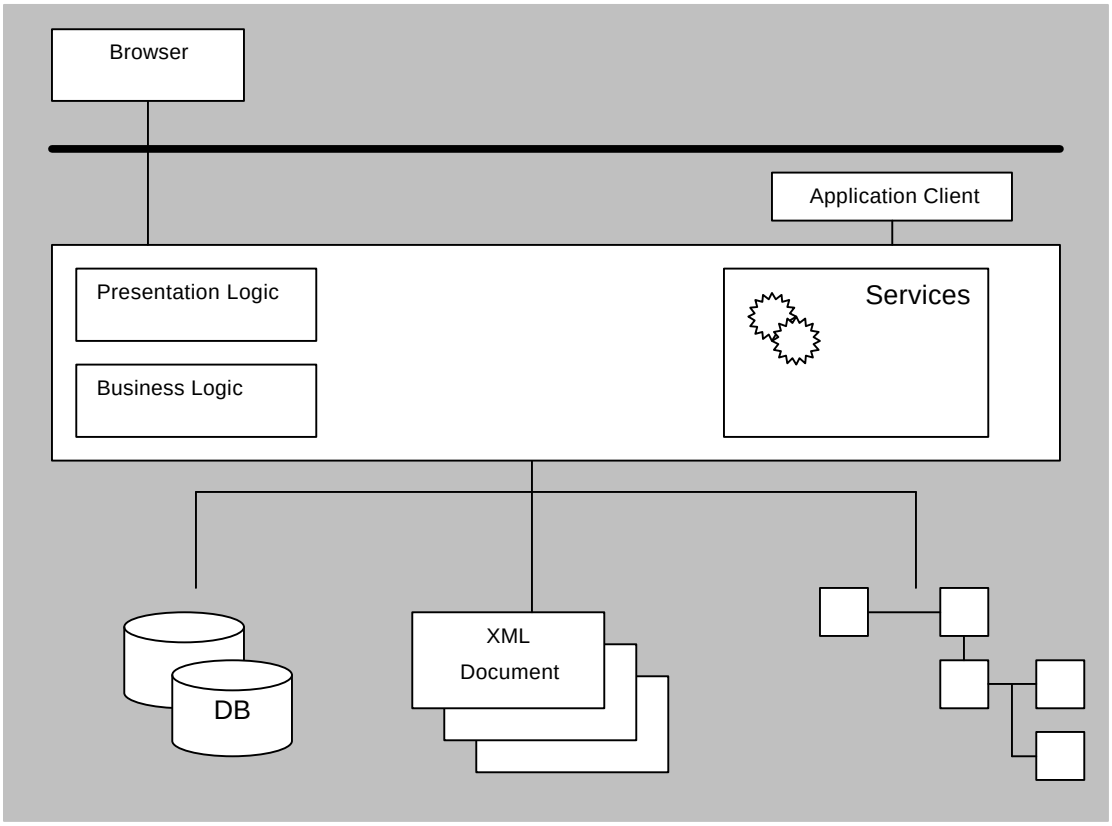
?? User Interface – Normalmente costituito da una interfaccia utente, ad esempio un web browser, che gira attraverso un firewall oppure una applicazione a finestre per Windows.

?? Presentation Logic – Definisce ciò che l'interfaccia utente deve mostrare e come devono essere soddisfatte le richieste dell'utente.

?? Business Logic – Modella i ruoli della applicazione, spesso attraverso l'interazione con lo schema dei dati.

?? Infrastructure Service – Sono funzionalità aggiuntive, necessarie da alcuni componenti della applicazione, ad esempio un supporto alle transazioni.

?? Data Layer – La fonte di immagazzinamento dei dati.



Enterprise Architecture

L'architettura di una applicazione Enterprise, può comprendere numerosi aspetti della applicazione che non sempre comunicano tra loro. Una architettura enterprise tuttavia è basata su un sistema n-tier. Per trasformare una sistema n-tier in uno enterprise è sufficiente estendere lo strato di mezzo dividendolo in più applicazion object che lavorano attraverso interfacce.

Con una architettura di tipo enterprise, è possibile avere un buon numero di applicazioni che utilizzano dei componenti in comune. Questo ovviamente porta ad una standardizzazione nel realizzare questi componenti.

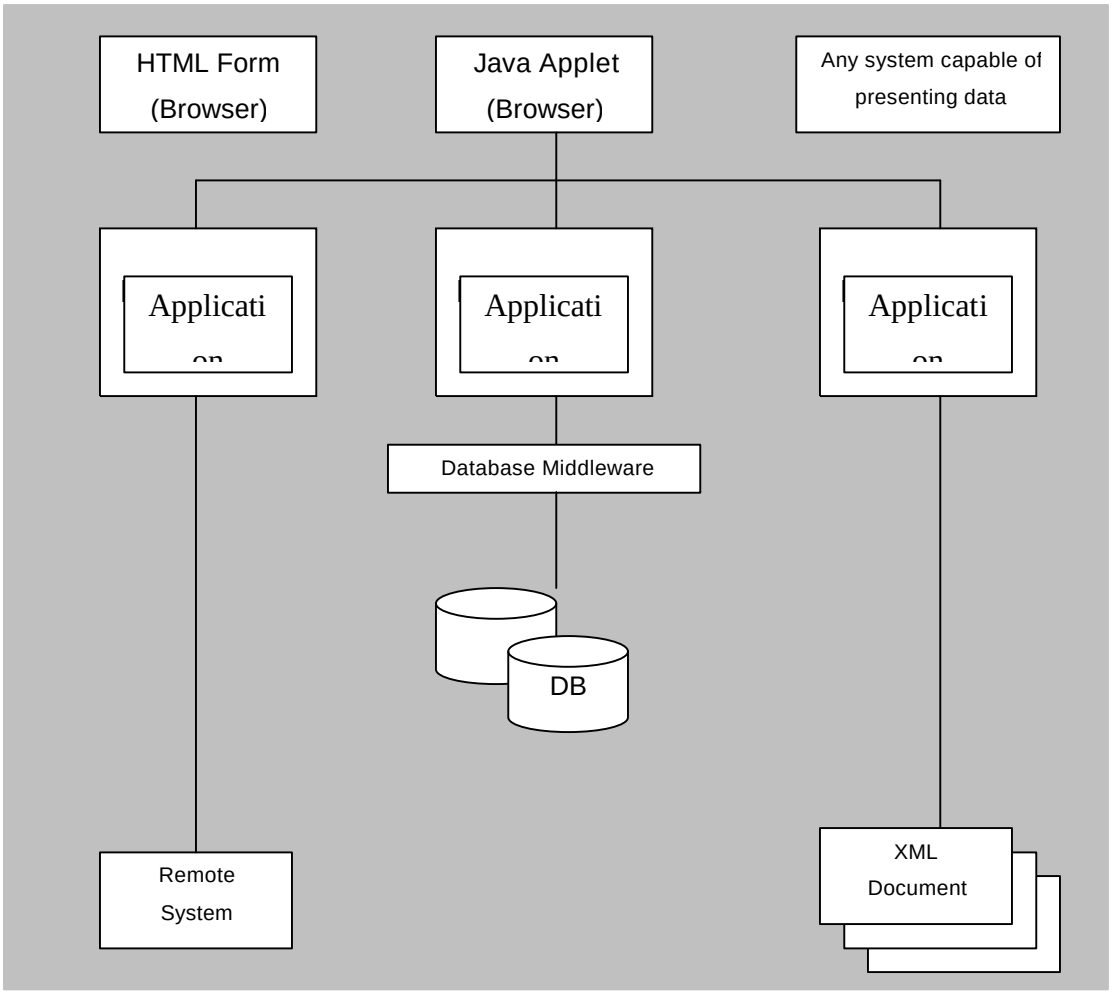


Figura A.5: Esempio di Enterprise Architecture

Java 2 Enterprise Edition API

Per realizzare applicazioni distribuite con Java è necessario accedere ad un insieme di servizi. Normalmente questi servizi includono sistemi per processare transazioni, messaggi, oppure, per la connettività ai database. L'architettura realizzata attraverso J2EE unifica l'accesso a cuesti servizi nella enterprise service API.

Le specifiche di J2EE definiscono un insieme standard di estensioni per Java.

?? JDBC 2.0 Extension (anche chiamato JDBC 2.0 Optional Package). - Queste API estendono quelle standard introducendo ad esempio il supporto per le transazioni distribuite.

?? Remote Method Invocation over the Internet Inter-ORB Protocol (RMI-IIOP) 1.0. - Fornisce una implementazione delle RMI attraverso IIOP. Questo colma la differenza tra le applicazioni RMI e Corba.

?? Enterprise Java Beans (EJB) 1.1. - Specifica un componente framwork per le applicazioni multi tier distribuite.

?? Java Servlets 2.2. - Le java servlet API servono per la realizzazioni di applicazioni web dinamiche.

?? Java Sever Page (JSP) 1.1. - Permettono la realizzazioni di pagine dinamiche, mesco lando l'uso di HTML e Java.

?? Java Message Service (JMS) 1.0

?? Java Naming and Directory Interface (JNDI) 1.2. - L'insieme di API che costituiscono JNDI vuole standardizzare l'accesso ai servizi di naming disponibili attualmente. In altre parole consente l'interrogazione di server LDAP o simili.

?? Java Transaction API 1.0. - Queste API servono per implementare applicazioni distribuite che sfruttano le transazioni.

?? JavaMail 1.1.- L'insieme di queste API permette di realizzare applicazioni indipendenti dalla piattaforma e dal protocollo per sfruttare le caratteristiche dei mail server.

Appendice B

Di seguito viene presentato per maggiore chiarezza lo schema E/R del progetto web della facoltà di Ingegneria Informatica.

Appendice C

Di seguito si riportano alcune delle parti maggiormente significative del codice relativo alla applicazione web realizzata. In particolare vengono trascritti i file necessari per visualizzare la lista dei docenti e la pagina web di un docente. Non vengono riportati tutti i file che si occupano di effettuare l'autentificazione di un utente nell'area protetta del sito web.

File di classe: Docente.java

```
/*
 * Docente.java
 *
 * Created on 6 marzo 2001, 11.33
 * Permette di ritrovare caratteristiche principali dalla tabella docenti
 */


/**
 *
 * @author  Luca Ciocci
 * @version 0.5.0
 */



import java.util.Vector;
import java.sql.*;
import javax.swing.table.AbstractTableModel;
import javax.swing.event.TableModelEvent;



public  class Docente {


    Connection          connection;
    Statement           statement;
    ResultSet           resultset;
    String[]            columnNames = {};
    Vector        rows = new Vector();
    ResultSetMetaData   metaData;

    public int ID;

    public String Nome;
    public String Cognome;
    public String Email;
    public String Tel;
    public String Fax;
    public String Ufficio;
    public String Link; //Non usato
    public String Ricevimento;
    public String Ricerca;
```

```
    public String Cooperazioni;
    public String Incarichi;
    public String Affiliazioni;
    public String Collaborazioni;
    public String Ruolo;

    public String Username; //Primary Key
    public String Password;



    /** Nel costruttore è necessario passare i parametri per la connessione
    /* al DB */
    public Docente(String url,String driverName,String user,String passwd)
    {
        try {
            Class.forName(driverName);
            System.out.println("Docente: Apertura della connessione al db");
            connection = DriverManager.getConnection(url, user, passwd);
            statement = connection.createStatement();
            }
            catch (ClassNotFoundException ex) {
                System.err.println("Docente Exception: Impossibile trovare la
classe del driver.");
                System.err.println(ex);
            }
            catch (SQLException ex) {
                System.err.println("Docente Exception: Impossibile  connettersi a
questo database.");
                System.err.println(ex);
            }


            try {
                resultset=statement.executeQuery("SELECT * FROM DOCENTE ORDER BY
USERNAME;");
                }
            catch (SQLException ex) {
                System.err.println("Docente Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }

     }

     public Docente(String url,String driverName,String  user,String
passwd,String sqlquery)
    {
        try {
            Class.forName(driverName);
            System.out.println("Docente: Apertura della connessione al db");
            connection = DriverManager.getConnection(url, user, passwd);
            statement = connection.createStatement();
            }
            catch (ClassNotFoundException ex) {
                System.err.println("Docente Exception: Impossibile trovare la
classe del driver.");
                System.err.println(ex);
            }
            catch (SQLException ex) {
                System.err.println("Docente Exception: Impossibile connettersi a
```

```
                System.err.println(ex);
            }


            try {
                resultset=statement.executeQuery(sqlquery);
                }
            catch (SQLException ex) {
                System.err.println("Docente Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }

     }

    /** Chiude l'oggetto perchè chiude la connessione. Effettivamente una
istanza
    della classe docente dopo questa operazione non è più utilizzabile*/
    public void close()
    {
            try {connection.close();}
            catch (SQLException ex) {
                System.err.println("Docente Exception: close: " + ex);
            }

    }

    public int Count()
    {//Restituisce il numero dei Docenti nel DB.  Effettua cioè un conteggio. Se
c'è un errore restituisce 0
      Statement tmpStat;
      ResultSet tmpRes;
      int       Num;


      try {
            Num=0;
            tmpStat = connection.createStatement();
            tmpRes=tmpStat.executeQuery("SELECT * FROM  DOCENTE;");
            while (tmpRes.next())
            {
                Num=Num+1;
            }

           }
      catch (SQLException ex)
          {
             return 0;
          }
      return Num;



    }

    public int GetFreeID()
    { //Restituisce un intero che rappresenta il primo valore disponibile di ID
      Statement tmpStat;
      ResultSet tmpRes;
      int       iFreeID;
```

```
      try {
            tmpStat = connection.createStatement();
            tmpRes=tmpStat.executeQuery("SELECT * FROM DOCENTE ORDER BY
IDDOCENTE DESC;");

            tmpRes.next();
            iFreeID=tmpRes.getInt("IDDOCENTE")+1;
            System.out.println("GetFr eeID:  " + iFreeID);
           }
      catch (SQLException ex)
          {
            System.out.println("GetFreeID exception: " + ex.toString());
            System.out.println("GetFreeID exception: " + iFreeID);
            return 0;
          }
       catch (java.lang.VerifyError ex) //Introdotta per compatibilita Interbase
Inteclient JDBC
          {
            System.out.println("GetFreeID exception: " + ex.toString());
            System.out.println("GetFreeID exception: " + iFreeID);
            return 0;
          }
      return iFreeID;
    }

    /** Questo metodo esegue la cancellazione di un record dalla tabella docente
    * in base all'username passato, si noti che username è PK
    * Ritorna 0 se la query ha successo
    *Ritornam 1 se c'è una SQLException
    */
    public int DeletebyUsername(String username)
    {

     String sDelete;
     sDelete="DELETE FROM DOCENTE WHERE USERNAME=?;";
     try {

          PreparedStatement DeleteStatement=
connection.prepareStatement(sDelete);
          DeleteStatement.setString(1,username);
          DeleteStatement.executeUpdate();
          DeleteStatement.close();
          connection.commit();

          return 0;
          }
     catch (SQLException sqle)
        {
           //Errore nell'eseguire la query di inserimento
           System.err.println("Docente Exception:DeletebyUsername: " + sqle);
           return 1;
        }
  }

    public int Insert(String username,String  passwd,String nome,St ring cognome)
    {

     String sInsert;
     sInsert="INSERT INTO DOCENTE(IDDOCENTE,USERNAME,PASSWD,NOME,COGNOME)
VALUES(?,?,?,?,?);";
```

```
     try {

          PreparedStatement InsertStatement=
connection.prepareStatement(sInsert);

          System.out.println("Insert: preparazione della query di inserimento");
          System.out.println(username);
          System.out.println(passwd);
          System.out.println(nome);
          System.out.println(cognome);
          System.out.println("-------------------------");

          //Verifico se è stato specificato l'ID della news
          //Altrimento lo creo io di default



          InsertStatement.setInt(1,GetFreeID());
          InsertStatement.setString(2,username);
          InsertStatement.setString(3,passwd);
          InsertStatement.setString(4,nome);
          InsertStatement.setString(5,cognome);

          System.out.println("Esecuzione della query");
          InsertStatement.executeUpdate();
          InsertStatement.close();
          connection.commit();
          System.out.println("Inserimento completato con successo");
          return 0;
         }

        catch (SQLException sqle)
        {
         //Errore nell'eseguire la query di inserimento
         System.err.println("Docente Exception:Insert: " + sqle);
         return 1;
        }

    }


    public int Update(String nome,String  cognome,String ufficio,
                        String tel, String fax, String email, String ricevimento
,
                          String ricerca , String cooperazioni , String
incarichi , String affiliazioni,String ruolo,
                               String whereUsername)
    {

     String sUpdate;
     sUpdate="UPDATE DOCENTE SET NOME=? , COGNOME=? , UFFICIO=? , TEL=? , FAX=?
, EMAIL=? , RICEVIMENTO=? , RICERCA=? , COOPERAZIONI=? , INCARICHI=? ,
AFFILIAZIONI=? , RUOLO=? WHERE USERNAME=? ;";


     try {

          PreparedStatement UpdateStatement=
connection.prepareStatement(sUpdate);

          System.out.println("Docente: Update: preparazione della query di
update");
```

```
          System.out.println("Nome: "+nome );
          System.out.println("Cognome: "+cognome);
          System.out.println("Ufficio: "+ufficio);
          System.out.println("Tel: "+tel);
          System.out.println("Fax: "+fax);
          System.out.println("Email: " + email);
          System.out.println("Ricevimento: "+ricevimento);
          System.out.println("Ricerca: " + ricerca);
          System.out.println("Cooperazioni: " + cooperazioni);
          System.out.println("Incarichi: " + incarichi);
          System.out.println("Affiliazioni: " + affiliazioni);
          System.out.println("Ruolo: " + ruolo);
          System.out.println("-------------------------");

          //Verifico se è stato specificato l'ID della news
          //Altrimento lo creo io di default

          UpdateStatement.setString(1,nome);
          UpdateStatement.setString(2,cognome);
          UpdateStatement.setString(3,ufficio);
          UpdateStatement.setString(4,tel);
          UpdateStatement.setString(5,fax);
          UpdateStatement.setString(6,email);
          UpdateStatement.setString(7,ricevimento);
          UpdateStatement.setString(8,ricerca);
          UpdateStatement.setString(9,cooperazioni);
          UpdateStatement.setString(10,incarichi);
          UpdateStatement.setString(11,affiliazioni);
          UpdateStatement.setString(12,ruolo);
          UpdateStatement.setString(13,whereUsername);

          System.out.println("Docente: Update: ");
          UpdateStatement.executeUpdate();
          UpdateStatement.close();
          connection.commit();
          System.out.println("Docente: Update: Aggiornamento completato con
successo");
           return 0;
         }

        catch (SQLException sqle)
        {
         //Errore nell'eseguire la query di inserimento
         System.err.println("Docente Exception: Update: " + sqle);
         return 1;
        }

    }
```

```
    public void MoveNext()
    { //Si sposta sulla news successiva
        try {
            resultset.next();
            }
        catch (SQLException ex)
```

```
        }
}

public void MovePrevious()
{//Si sposta sulla news precedente
    try {
        resultset.previous();
        }
    catch (SQLException ex)
        {

        }
}



public int GetID()
{//Restituisce una stringa contenete l'ID
    try{
       ID=resultset.getInt("IDDocente");
       }
   catch  (SQLException ex){}
   return ID;
}

public String GetUsername()
{//Restituisce una stringa contenente l'Username del docente
   try{
       Username=resultset.getString("USERNAME");
       System.out.println("Docente:GetUsername:" +Us ername);
      }
    catch  (SQLException ex)
    {
      System.out.println("Docente Exception:GetUsername: " +ex.toString());
    }
    return Username.toString();

}

public String GetPassword()
{//Restituisce una stringa contenete la descrizione della news
    try {
        Password=resultset.getString("PASSWD");
        System.out.println("Docente:GetPassword:" +Password);
        }
    catch  (SQLException ex)
    {
      System.out.println("Docente Exception:GetPassword: " +ex.toString());
    }
    return Password.toString();
}

public String GetNome()
{//Restituisce una stringa contenete il link
    try{
       Nome=resultset.getString("NOME");
       System.out.println("Docente:GetNome:" +Nome);
       }
    catch  (SQLException ex)
    {
    System.out.println("Docente Exception:GetNome: " +ex.toString());
```

```
    }
    return Nome.toString();
}
public String GetCognome()
{//Restituisce una stringa contenete il cognome
    try{
           Cognome=resultset.getString("COGNOME");
           System.out.println("Docente:GetCogome:" +Cognome);
       }
       catch  (SQLExc eption ex)
      {
        System.out.println("Docente Exception:GetCognome: " +ex.toString());
      }
    return Cognome.toString();
}

 public String GetEmail()
{//Restituisce una stringa contenente l'Email del docente
   try{
       Email=resultset.getString("EMAIL");
       System.out.println("Docente:GetEmail:" +Email);
      }
    catch  (SQLException ex)
    {
      System.out.println("Docente Exception:GetEmail: " +ex.toString());
    }

    if (Email==null) return "";
    return Email.toString();

}

public String GetTel()
{//Restituisce una stringa contenente il Telefono del docente
   try{
       Tel=resultset.getString("TEL");
       System.out.println("Docente:GetTel:" +Tel);
      }
    catch  (SQLException ex)
    {
      System.out.println("Docente Exception:GetTel: " +ex.toString());
    }
    if (Tel==null) return "";
    else return Tel.toStrin g();

}

public String GetFax()
{//Restituisce una stringa contenente il FAX del docente
   try{
       Fax=resultset.getString("FAX");
       System.out.println("Docente:GetFax:" +Fax);
      }
    catch  (SQLException ex)
    {
      System.out.println("Docente Exception:GetFax: " +ex.toString());
    }
    if (Fax==null) return "";
    else return Fax.toString();
}
```

```
    {//Restituisce una stringa contenente l'ubicazione dell'ufficio del docente
       try{
           Ufficio=resultset.getString("UFFICIO");
           System.out.println("Docente:GetUfficio:" +Ufficio);
         }
        catch  (SQLException ex)
        {
          System.out.println("Docente Exception:GetUfficio: " +ex.toString());
        }

        if (Ufficio==null) return "";
        else return Ufficio.toString();
    }


    /**Non utilizzato */
    public String GetLink()
    {
       try{
           Link=resultset.getString("LINK");
           System.out.println("Docente:GetLink:" +Link);
         }
        catch  (SQLException ex)
        {
          System.out.println("Docente Exception:GetLink: " +ex.toString());
        }
        if (Link==null) return "";
        else return Link.toString();
    }

    public String GetRicevimento()
    {//Restituisce una stringa contenente l'orario e il luogo di ricevimento
docente
       try{
           Ricevimento=resultset.getString("RICEVIMENTO");
           System.out.println("Docente:GetRicevimento:" +Ricevimento);
           }

        catch  (SQLException ex)
        {
          System.out.println("Docente Exception:GetRicevimento: "
+ex.toString());
          return "";
        }

        if (Ricevimento==null) return "";
        else return Ricevimento.toString();
    }

    public String GetRicerca()
    {//Restituisce una stringa contenente l'orario e il luogo di ricevimento
docente
       try{
           Ricerca=resultset.getString("RICERCA");
           System.out.println("Docente:GetRicerca:" +Ricevimento);
         }
        catch  (SQLException ex)
        {
          System.out.println("Docente Exception:GetRicerca: " +ex.toString());
        }
        if (Ricerca==null) return "";
        else return Ricerca.toString();
```

```
    public String GetCooperazioni()
    {//Restituisce una stringa
       try{
           Cooperazioni=resultset.getString("COOPERAZIONI");
           System.out.println("Docente:GetCoo perazioni:" +Cooperazioni);
         }
        catch  (SQLException ex)
        {
          System.out.println("Docente Exception:GetCooperazioni: "
+ex.toString());
        }

        if (Cooperazioni==null) return "";
        else return Cooperazioni.toString();
    }

    public String GetIncarichi()
    {//Restituisce una stringa
       try{
           Incarichi=resultset.getString("INCARICHI");
           System.out.println("Docente:GetIncarichi:" +Incarichi);
         }
        catch  (SQLException ex)
        {
          System.out.println("Docente Exception:GetIncarichi: " +ex.toString());
        }
        if (Incarichi==null) return "";
        else return Incarichi.toString();
    }

    public String GetAffiliazioni()
    {//Restituisce una stringa
       try{
           Affiliazioni=resultset.getString("AFFILIAZIONI");
           System.out.println("Docente:GetAffiliazioni:" +Affiliazioni);
         }
        catch  (SQLException ex)
        {
          System.out.println("Do cente Exception:GetAffiliazioni: "
+ex.toString());
        }
        if (Affiliazioni==null) return "";
        else return Affiliazioni.toString();
    }

    public String GetCollaborazioni()
    {//Restituisce una stringa
       try{
           Collaborazioni=resultset.getString("COLLABORAZIONI");
           System.out.println("Docente:GetCollaborazioni:" +Collaborazioni);
         }
        catch  (SQLException ex)
        {
          System.out.println("Docente Exception:GetCollaborazioni: "
+ex.toString());
        }
        if (Collaborazioni==null) return "";
        else return Collaborazioni.toString();
    }

    public String GetRuolo()
```

```
        try{
               Ruolo=resultset.getString("RUOLO");
               System.out.println("Docente: GetRuolo:" +Ruolo);
            }
            catch  (SQLException ex)
           {
             System.out.println("Docente Exception: GetRuolo: " +ex.toString());
           }

        if (Ruolo==null) return "";
        else return Ruolo.toString();
    }
}
```

File di classe: Insegnamento.java

```
/*
 * Insegnamento.java
 *
 * Created on 6 marzo 2001, 11.33
 * Permette di ritrovare caratterisitche principali dalla tabella Insegnam ento
 */


/**
 *
 * @author  l.ciocci
 * @version 0.5.0
 */



import java.util.Vector;
import java.sql.*;
import javax.swing.table.AbstractTableModel;
import javax.swing.event.TableModelEvent;




public class Insegnamento {


    Connection          connection;
    Statement           statement;
    ResultSet           resultset;
    String[]            columnNames = {};
    Vector        rows = new Vector();
    ResultSetMetaData   metaData;

    public int ID; //Primary Key

    public String Nome; //Nome dell'Insegnamento es: Basi di Dati
    public String Descrizione;
    public String Programma;
    public String ProgPrecedente;
```

```
    private int IDDocente;        // References Docente
    private int IDInsegnamento;   // Primary
    //---------------

    /** Nel costruttore è necessario passare i paramtri per la connessione al DB
necessario passare i parametri di connessione al db */
    public Insegnamento(String url,String driverName,String user,String passwd,
int iddocente)
    {

      IDDocente=iddocente;

      try {
            Class.forName(driverName);
            System.out.println("Insegnamento: Apertura della connessione al
db");
            connection = DriverManager.getConnection(url, user, passwd);
            statement = connection.createStatement();
            }
            catch (ClassNotFoundException ex) {
                System.err.println("Insegnamento Exception: Impossibile trovare
la classe del driver.");
                System.err.println(ex);
            }
            catch (SQLException ex) {
                System.err.println("Insegnamento Exception: Impossibile
connettersi a questo database.");
                System.err.println(ex);
            }


            try {
                resultset=statement.executeQuery("SELECT * FROM INSEGNAMENTO
WHERE IDDOCENTE="+ IDDocente + " ORDER BY IDINSEGNAMENTO;");
                }
            catch (SQLException ex) {
                System.err.println("Insegnamento Exception: Impo ssibile
connettersi a questo database.");
                System.err.println(ex);
            }

     }

     public Insegnamento(String url,String driverName,String user,String
passwd,String sqlquery,int iddocente)
    {

      IDDocente=iddocente;
      try {
            Class.forName(driverName);
            System.out.println("Insegnamento: Apertura della connessione al
db");
            connection = DriverManager.getConnection(url, user, passwd);
            statement = connection.createStatement();
            }
            catch (ClassNotFoundException ex) {
                System.err.println("Insegnamento Exception: Impossibile trovare
la classe del driver.");
                System.err.println(ex);
            }
            catch (SQLException ex) {
                System.err.println("Insegnamento Exception: Impossibile
```

```
                System.err.println(ex);
            }


            try {
                resultset=statement.executeQuery(sqlquery);
                }
            catch (SQLException ex) {
                System.err.println("Docente Insegnamento: Impossibile
connettersi a questo database.");
                System.err.println(ex);
            }

     }


    public Insegnamento(int idinsegnamento,String url,String driverName,String
user,String passwd)
    {

      try {
            Class.forName(driverName);
            System.out.println("Insegnamento: Apertura della connessione al
db");
            connection = DriverManager.getConnection(url, user, passwd);
            statement = connection.createStatement();
            }
            catch (ClassNotFoundException ex) {
                System.err.println("Insegnamento Exception: Impossibile trovare
la classe del driver.");
                System.err.println(ex);
            }
            catch (SQLException ex) {
                System.err.println("Insegnamento Exception: Impossibile
connettersi a questo database.");
                System.err.println(ex);
            }


            try {
                resultset=statement.executeQuery("SELECT * FROM INSEGNAMENTO
WHERE IDINSEGNAMENTO="+idinsegnamento+";");
                IDDocente=GetIDDocente();

                }
            catch (SQLException ex) {
                System.err.println("Docente Insegnamento: Impossibile
connettersi a questo database.");
                System.err.println(ex);
            }
     }

    /** Chiude l'oggetto perchè chiude la connessione. Effettivamente una
istanza
    della classe Insegnamento dopo questa operazione non è più utilizzabile*/
    public void close()
    {
            try {connection.close();}
            catch (SQLException ex) {
                System.err.println("Insegnamento Exception: close: " + ex);
            }
```

```
    public int Count()
    {//Restituisce il numero di Insegnamenti nel DB. Effettua cioè un conteggio.
Se c'è un errore restituisce 0
      Statement tmpStat;
      ResultSet tmpRes;
      int       Num;


      try {
            Num=0;
            tmpStat = connection.createStatement();
            tmpRes=tmpStat.executeQuery("SELECT * FROM INSEGNAMENTO WHERE
IDDOCENTE="+IDDocente+";");
            while (tmpRes.next())
            {
                Num=Num+1;
            }

           }
      catch (SQLException ex)
          {
             return 0;
          }
      return Num;



    }

   /**Restituisce un intero che rappresenta il primo valore disponibile di ID
<BR>
       Probabilmente necessita di una revisione!!!  */
    public int GetFreeIDInsegnamento()
    {
      Statement tmpStat;
      ResultSet tmpRes;
      int       iFreeID;


      iFreeID=0;

      try {
            System.out.println("Insegnamento: GetFreeIDInsegnamento: Apertura
recordset" );
            tmpStat = connection.createStatement();
            tmpRes=tmpStat.executeQuery("SELECT * FROM INSEGNAMENTO ORDER BY
IDINSEGNAMENTO DESC;");

            tmpRes.next();
            iFreeID=tmpRes.getInt("IDINSEGNAMENTO")+1;
            System.out.println("Insegnamento: GetFreeIDInsegnamento:  " +
iFreeID);

           }
      catch (SQLException ex)
          {
            System.out.println("Insegnamento: GetFreeID exception: " +
ex.toString());
            System.out.println("Insegnamento: GetFreeID exception: " + iFreeID);
            return 0;
          }
```

```
      catch (java.lang.VerifyError ex) //Introdotta per compatibilita Interbase
Inteclient JDBC
          {
            System.out.println("GetFreeID exception: " + ex.toString());
            System.out.println("GetFreeID exception: " + iFreeID);
            return 0;
          }
      return iFreeID;

    }

    /** Questo metodo esegue la cancellazione di un record dalla tabella
Insegnamento
    * in base all'IDInsegnamento passato, si noti che IDInsegnamento e IDDOcente
sono PK <BR>
    * Ritorna 0 se la query ha successo <BR>
    * Ritorna 1 se c'è una SQLException
    */
    public int Delete(int idinsegnamento)
    {

     String sDelete;
     sDelete="DELETE FROM INSEGNAMENTO WHERE IDINSEGNAMENTO=?;";
     try {

          PreparedStatement DeleteStatement=
connection.prepareStatement(sDelete);
          DeleteStatement.setInt(1,idinsegnamento);
          DeleteStatement.executeUpdate();
          DeleteStatement.close();
          connection.commit();

          return 0;
          }
     catch (SQLException sqle)
        {
           //Errore nell'eseguire la query di inserimento
           System.err.println("Insegnamento Exception:DeletebyUsername: " +
sqle);
           return 1;
        }
  }

    public int Insert(String nome,String descrizione,String programma)
    {

     String sInsert;
     sInsert="INSERT INTO
INSEGNAMENTO(IDDOCENTE,IDINSEGNAMENTO,NOME,DESCRIZIONE,PROGRAMMA)
VALUES(?,?,?,?,?);";


     try {

          PreparedStatement InsertStatement=
connection.prepareStatement(sInsert);

          System.out.println("Insegnamento: Insert: preparazione della query di
inserimento");
          System.out.println(nome);
          System.out.println(descrizione);
          System.out.println(programma);
```

```
          //Verifico se è stato specificato l'ID della news
          //Altrimento lo creo io di default

          GetFreeIDInsegnamento();

          InsertStatement.setInt(1,IDDocente);
          InsertStatement.setInt(2,GetFreeIDInsegnamento());
          InsertStatement.setString(3,nome);
          InsertStatement.setString(4,descrizione);
          InsertStatement.setString(5,programma);

          System.out.println("Insegnamento: Esecuzione della query");
          InsertStatement.executeUpdate();
          InsertStatement.close();
          connection.commit();
          System.out.println("Insegnamento: Inserimento completato con
successo");
           return 0;
         }

        catch (SQLException sqle)
        {
         //Errore nell'eseguire la query di inserimento
         System.err.println("Insegnamento Exception :Insert: " + sqle);
         return 1;
        }

    }


    public int Update(String nome,String  descrizione,String programma)
    {

     String sUpdate;
     sUpdate="UPDATE INSEGNAMENTO SET NOME=? , DESCRIZIONE=? , PROGRAMMA=?  WHERE
IDDOCENTE=? AND IDINSEGNAMENTO=?;";


     try {

          PreparedStatement UpdateStatement=
connection.prepareStatement(sUpdate);

          System.out.println("Insegnamento: Update: preparazione della query  di
update");

          System.out.println("Nome: "+nome);
          System.out.println("Descrizione: "+descrizione);
          System.out.println("Programma: "+programma);
          System.out.println("IDDocente: " + IDDocente);
          System.out.println("IDInsegnamento: " + IDInsegnamento);
          System.out.println("-------------------------");



          UpdateStatement.setString(1,nome);
          UpdateStatement.setString(2,descrizione);
          UpdateStatement.setString(3,programma);
          UpdateStatement.setInt(4,IDDocente);
          UpdateStatement.setInt(5,IDInsegnamento);
```

```
          UpdateStatement.executeUpdate();
          UpdateStatement.close();
          connection.commit();
          System.out.println("Insegnamento: Update: Aggiornamento completato con
successo");
          return 0;
         }

        catch (SQLException sqle)
        {
         //Errore nell'eseguire la query di inserimento
         System.err.println("Insegnamento Exception: Update: " + sqle);
         return 1;
        }

    }


    public int Update(String nome,String  descrizione,String programma, int
idinsegnamento)
    {

     String sUpdate;
     sUpdate="UPDATE INSEGNAMENTO SET NOME=? , DESCRIZIONE=? , PROGRAMMA=?  WHERE
IDINSEGNAMENTO=?;";
     try {

          PreparedStatement UpdateStatement=
connection.prepareStatement(sUpdate);

          System.out.println("Insegnamento: Update: preparazione della query di
update");

          System.out.println("Nome: "+nome);
          System.out.println("Descrizione: "+descrizione);
          System.out.println("Programma: "+progr amma);
          System.out.println("IDDocente: " + IDDocente);
          System.out.println("IDInsegnamento: " + idinsegnamento);
          System.out.println("------------------------");



          UpdateStatement.setString(1,nome);
          UpdateStatement.setString(2,descrizione);
          UpdateStatement.setString(3,programma);
          UpdateStatement.setInt(4,idinsegnamento);

          System.out.println("Insegnamento: Update: ");
          UpdateStatement.executeUpdate();
          UpdateStatement.close();
          connection.commit();
          System.out.println("Insegnamento: Update: Aggiornamento completato con
successo");
          return 0;
         }

        catch (SQLException sqle)
        {
         //Errore nell'eseguire la query di inserimento
         System.err.println("Insegnamento Exception: Update: " + sqle);
         return 1;
        }
```

```
    }


    public void MoveNext()
    { //Si sposta sulla news successiva
        try {
            resultset.next();
            }
        catch (SQLException ex)
            {

            }
    }

    public void MovePrevious()
    {//Si sposta sulla news precedente
        try {
            resultset.previous();
            }
        catch (SQLException ex)
            {

            }
    }



    public int GetIDDocente()
    {//Restituisce una stringa contenete l'ID
        try{
           IDDocente=resultset.getInt("IDDOCENTE");
           }
       catch  (SQLException ex){}

       catch (java.lang.VerifyError ex) //Introdotta per compatibilita Interbase
Inteclient JDBC
          {

            return 0;
          }
       return IDDocente;
    }

    public int GetIDInsegnamento()
    {//Restituisce una stringa contenete l'ID
        try{
           IDInsegnamento=resultset.getInt("IDInsegnamento");
           }
       catch  (SQLException ex){}
       return IDInsegnamento;
    }


    public String GetNome()
    {//Restituisce una stringa contenete il Nome dell'insegnamento
        try{
           Nome=resultset.getString("NOME");
           System.out.println("Insegnamento: GetNome:" +Nome);
           }
        catch  (SQLException ex)
        {
        System.out.println("Insegnamento Exception: GetNome: " +ex.toString());
```

```
        }
        return Nome.toString();
    }

     public String GetDescrizione()
    {//Restituisce una stringa contenente la descrizione dell'insegnamento
       try{
           Descrizione=resultset.getString("DESCRIZIONE");
           System.out.println("Insegnamento: GetDescrizione" +Descrizione);
         }
        catch  (SQLException ex)
        {
          System.out.println("Insegnamento Exception: Ge tDescrizione: "
+ex.toString());
        }

        if (Descrizione==null) return "";
        return Descrizione.toString();

    }

    public String GetProgramma()
    {//Restituisce una stringa contenente il Telefono del docente
       try{
           Programma=resultset.getString("PROGRAMMA");
           System.out.println("Insegnamento: GetProgramma:" +Programma);
         }
        catch  (SQLException ex)
        {
          System.out.println("Insegnamento Exception: GetProg ramma: "
+ex.toString());
        }
        if (Programma==null) return "";
        else return Programma.toString();

    }


    public String GetProgPrecedente()
    {//Restituisce una stringa contenente il Telefono del docente
       try{
           ProgPrecedente=resultset.getString("PROGPRECEDENTE");
           System.out.println("Insegnamento: GetProgPrecedente:"
+ProgPrecedente);
         }
        catch  (SQLException ex)
        {
          System.out.println("Insegnamento Exc eption: GetProgPrecedente: "
+ex.toString());
        }
        if (ProgPrecedente==null) return "";
        else return ProgPrecedente.toString();

    }

}
```

File di classe: Appello.java

```
/*
 * Appello.java
 *
 * Created on 6 marzo 2001, 11.33
 * Permette di ritrovare caratterisitche principali dalla tabella Appello
 */


/**
 *
 * @author  l.ciocci
 * @version 0.5.0
 */



import java.util.Vector;
import java.sql.*;
import javax.swing.table.AbstractTableModel;
import javax.swing.event.TableModelEvent;




public class Appello {


    Connection          connection;
    Statement           statement;
    ResultSet           resultset;
    String[]            columnNames = {};
    Vector        rows = new Vector();
    ResultSetMetaData   metaData;

    public int IDInsegnamento;


    public String Data;
    public String Ora;
    public String Aula;
    public String Descrizione;

    /** Nel costruttore è necessario passare i paramtri per la connessione al DB
necessario passare i parametri di connessione al db */
    public Appello(String url,String driverName,String user,String passwd)
    {
        try {
            Class.forName(driverName);
            System.out.println("Appello: Apertura della connessione al db");
            connection = DriverManager.getConnection(url, user, passwd);
            statement = connection.createStatement();
            }
            catch (ClassNotFoundException ex) {
                System.err.println("AppelloException: Impossibile t rovare la
classe del driver.");
                System.err.println(ex);
            }
```

```
                System.err.println("Appello Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }


            try {
                resultset=statement.executeQuery("SELECT * FROM APPELLO ORDER BY
IDINSEGNAMENTO,DATA;");
                }
            catch (SQLException ex) {
                System.err.println("Appello Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }

     }

     public Appello(String url,String driverName,String user,String
passwd,String sqlquery)
    {
        try {
            Class.forName(driverName);
            System.out.println("Appello: Apertura della connessione al db");
            connection = DriverManager.getConnection(url, user, passwd);
            statement = connection.createStatement();
            }
            catch (ClassNotFoundException ex) {
                System.err.println("Appello Exception: Impossibile trovare la
classe del driver.");
                System.err.println(ex);
            }
            catch (SQLException ex) {
                System.err.println("Appello Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }


            try {
                resultset=statement.executeQuery(sqlquery);
                }
            catch (SQLException ex) {
                System.err.println("Appello Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }

     }

    /** Chiude l'oggetto perchè chiude la connessione. Effettivamente una
istanza
    della classe docente dopo questa operazione non è più utilizzabile*/
    public void close()
    {
            try {connection.close();}
            catch (SQLException ex) {
                System.err.println("Appello Exception: close: " + ex);
            }

    }
```

```
    /**Restituisce il numero degli appelli nel DB. <BR>
       Effettua cioè un conteggio. <BR>
       Se c'è un errore restituisce 0
    */
    public int Count(int idinsegnamento)
    {
      Statement tmpStat;
      ResultSet tmpRes;
      int       Num;


      try {
            Num=0;
            tmpStat = connection.createStatement();
            tmpRes=tmpStat.executeQuery("SELEC T * FROM APPELLO WHERE
IDINSEGNAMENTO=" + idinsegnamento +";");
            while (tmpRes.next())
            {
                Num=Num+1;
            }

           }
      catch (SQLException ex)
          {
             return 0;
          }
      return Num;
    }

    /** Esegue la cancellazone di un appello nella data specificata come
parametro
        e con il relativo idinsegnamento */
    public int Delete(String dataAppello, int idinsegnamento)
    {

     String sDelete;
     sDelete="DELETE FROM APPELLO WHERE DATA=? AND IDINSEGNAMENTO=?";
     try {

          PreparedStatement DeleteStatement=
connection.prepareStatement(sDelete);

          DeleteStatement.setDate(1,Date.valueOf(dataAp pello));
          DeleteStatement.setInt(2,idinsegnamento);
          DeleteStatement.executeUpdate();
          DeleteStatement.close();
          connection.commit();

          return 0;
          }
     catch (SQLException sqle)
        {
           //Errore nell'eseguire la query di cancellazione
           System.err.println("Appello Exception: Delete: " + sqle);
           return 1;
        }
  }

    public int Insert(int idinsegnamento, String dataAppello,String  ora,String
aula,String descrizione)
    {
```

```
     String sInsert;
     sInsert="INSERT INTO APPELLO(IDINSEGNAMENTO,DATA,ORA,AULA,DESCRIZIONE)
VALUES(?,?,?,?,?);";


     try {

          PreparedStatement InsertStatement=
connection.prepareStatement(sInsert);

          System.out.println("Appello: Insert: preparazione della query di
inserimento");
          System.out.println(dataAppello);
          System.out.println(ora);
          System.out.println(aula);
          System.out.println(descrizione);
          System.out.println("------------------------");


          InsertStatement.setInt(1,idinsegnamento);
          InsertStatement.setDate(2,Date.valueOf(dataAppello));
          InsertStatement.setString(3,ora);
          InsertStatement.setString(4,aula);
          InsertStatement.setString(5,descrizione);

          System.out.println("Esecuzione della query");
          InsertStatement.executeUpdate();
          InsertStatement.close();
          connection.commit();
          System.out.println("Appello: Insert: Inserimento completato con
successo");
           return 0;
         }

        catch (SQLException sqle)
        {
         //Errore nell'eseguire la query di inserimento
         System.err.println("Appello Exception:Insert: " + sqle);
         return 1;
        }

    }


    public int Update(int idinsegnamento, String dataAppello,String  ora,String
aula,String descrizione, int whereIDInsegnamento, String  whereData)
    {

     String sUpdate;
     sUpdate="UPDATE APPELLO SET IDINSEGNAMENTO=? , DATA=? , ORA=? , AULA=? ,
DESCRIZIONE=? , WHERE IDINSEGNAMENTO=? AND DATA=?;";


     try {

          PreparedStatement UpdateStatement=
connection.prepareStatement(sUpdate);

          System.out.println("Appello: Update: preparazione della query di
update");

          System.out.println("IDInsegnamento: "+idinsegnamento);
```

```
          System.out.println("Ora: "+ora);
          System.out.println("Aula: "+aula);
          System.out.println("Descrizione: " + descrizione);
          System.out.println("------------------------");

          //Verifico se è stato specificato l'ID della news
          //Altrimento lo creo io di default

          UpdateStatement.setInt(1,idinsegnamento);
          UpdateStatement.setDate(2,Date.valueOf(dataAppello));
          UpdateStatement.setString(3,ora);
          UpdateStatement.setString(4,aula);
          UpdateStatement.setString(5,descrizione);
          UpdateStatement.setInt(6,whereIDInsegnamento);
          UpdateStatement.setDate(7,Date.valueOf(whereData));
          System.out.println("Appello: Update: ");
          UpdateStatement.executeUpdate();
          UpdateStatement.close();
          connection.commit();
          System.out.println("Appello: Update: Aggiornamento completato con
successo");
          return 0;
         }

        catch (SQLException sqle)
        {
         //Errore nell'eseguire la query di inserimento
         System.err.println("Appello Exception: Update: " + sqle);
         return 1;
        }

    }

    public void MoveNext()
    { //Si sposta sulla news successiva
        try {
            resultset.next();
            }
        catch (SQLException ex)
            {

            }
    }

    public void MovePrevious()
    {//Si sposta sulla news precedente
        try {
            resultset.previous();
            }
        catch (SQLException ex)
            {

            }
    }




    public String GetData()
    {//Restituisce una stringa contenete il link
        try{
           Data=resultset.getString("DATA");
```

```
        }
    catch  (SQLException ex)
    {
    System.out.println("Appello Exception: GetData: " +ex.toString());

    }
    return Data.toString();
}

public Date GetObjData()
{//Restituisce una stringa contenete il link
    try{
       java.sql.Date ObjData=resultset.getDate("DATA");
       System.out.println("Appello: GetObjData:" +ObjData.to String());
       return ObjData;
       }
    catch  (SQLException ex)
    {
    System.out.println("Appello Exception: GetObjData: " +ex.toString());
    return null;
    }
}


public String GetOra()
{//Restituisce una stringa contenete il cognome
    try{
           Ora=resultset.getString("ORA");
           System.out.println("Appello: GetOra:" +Ora);
       }
       catch  (SQLException ex)
      {
        System.out.println("Appello Exception: GetOra: " +ex.toString());
      }
    return Ora.toString();
}

 public String GetAula()
{//Restituisce una stringa contenente l'Email del docente
   try{
       Aula=resultset.getString("AULA");
       System.out.println("Appello: GetAula:" +Aula);
      }
    catch  (SQLException ex)
    {
      System.out.println("Appello Exception: GetAula: " +ex.toString());
    }

    if (Aula==null) return "";
    return Aula.toString();

}

public String GetDescrizione()
{//Restituisce una stringa contenente il Telefono del docente
   try{
       Descrizione=resultset.getString("DESCRIZIONE");
       System.out.println("Appello: GetDescrizione: " +Descrizione);
      }
    catch  (SQLException ex)
    {
      System.out.println("Appello Exception: GetDescrizione: "
```

```
        }
        if (Descrizione==null) return "";
        else return Descrizione.toString();

    }
}
```

File di Classe: Avviso.java

```
*
 * Created on 6 marzo 2001, 11.33
 * Permette di ritrovare caratterisitche principali dalla tabella Avviso
 */


/**
 *
 * @author  l.ciocci
 * @version 0.5.0
 */



import java.util.Vector;
import java.sql.*;
import javax.swing.table.AbstractTableModel;
import javax.swing.event.TableModelEvent;



public class Avviso {


    Connection          connection;
    Statement           statement;
    ResultSet           resultset;
    String[]            columnNames = {};
    Vector        rows = new Vector();
    ResultSetMetaData   metaData;

    public int IDInsegnamento; //references INSEGNAMENTO
    public int IDAvviso;       //Primary Key


    public String Data;
    public String Autore;
    public String Aula;
    public String TitoloI;
    public String TitoloE;  //Unused
    public String DescrizioneI;
    public String DescrizioneE;  //Unused

    /** Nel costruttore è necessario passare i paramtri per  la connessione al DB
necessario passare i parametri di connessione al db */
    public Avviso(String url,String driverName,String user,String passwd)
```

```
    {
        try {
            Class.forName(driverName);
            System.out.println("Avviso: Apertura della connessione al db");
            connection = DriverManager.getConnection(url, user, passwd);
            statement = connection.createStatement();
            }
            catch (ClassNotFoundException ex) {
                System.err.println("Avviso Exception: Impossibile trovare la
classe del driver.");
                System.err.println(ex);
            }
            catch (SQLException ex) {
                System.err.println("Avviso Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }


            try {
                resultset=statement.executeQuery("SELECT * FROM AVVISO ORDER BY
IDINSEGNAMENTO,DATA;");
                }
            catch (SQLException ex) {
                System.err.println("Avviso Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }
            catch (java.lang.VerifyError ex) //Introdotta per compatibilita
Interbase Inteclient JDBC
            {
              System.out.println("Mah!" + ex.toString());

            }

     }

     public Avviso(String url,String driverName,String user,String passwd,String
sqlquery)
    {
        try {
            Class.forName(driverName);
            System.out.println("Avviso: Apertura della connessione al db");
            connection = DriverManager.getConnection(url, user, passwd);
            statement = connection.createStatement();
            }
            catch (ClassNotFoundException ex) {
                System.err.println("Avviso Exception: Impossibile trovare la
classe del driver.");
                System.err.println(ex);
            }
            catch (SQLException ex) {
                System.err.println("Avviso Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }
            catch (java.lang.VerifyError ex) //Introdotta per compatibilita
Interbase Inteclient JDBC
            {
              System.out.println("Mah!" + ex.toString());

            }
```

```
            try {
                resultset=statement.executeQuery(sqlquery);
                }
            catch (SQLException ex) {
                System.err.println("Avviso  Exception: Impossibile connettersi a
questo database.");
                System.err.println(ex);
            }
            catch (java.lang.VerifyError ex) //Introdotta per compatibilita
Interbase Inteclient JDBC
            {
              System.out.println("Mah!" + ex.toString());

            }

     }

    /** Chiude l'oggetto perchè chiude la connessione. Effettivamente una
istanza
    della classe docente dopo questa operazione non è più utilizzabile*/
    public void close()
    {
            try {connection.close();}
            catch (SQLException ex) {
                System.err.println("Avviso Exception: close: " + ex);
            }

    }

    /**Restituisce il numero degli avvisi per insegnamento nel DB.
        Effettua cioè un conteggio. Se c'è un errore restituisce 0
    */

    public int Count(int idinsegnamento)
    {
      Statement tmpStat;
      ResultSet tmpRes;
      int       Num;


      try {
            Num=0;
            tmpStat = connection.createStatement();
            tmpRes=tmpStat.executeQuery("SELECT * FROM AVVISO WHERE
IDINSEGNAMENTO=" + idinsegnamento +";");
            while (tmpRes.next())
            {
                Num=Num+1;
            }
           }
      catch (SQLException ex)
          {
             return 0;
          }

      catch (java.lang.VerifyError ex) //Introdotta per compatibilita Interbase
Inteclient JDBC
            {
              return 0;
            }
      return Num;
    }
```

```
    public int Delete(int idavviso)
    {

     String sDelete;
     sDelete="DELETE FROM AVVISO WHERE IDAVVISO=?";
     try {

          PreparedStatement DeleteStatement=
connection.prepareStatement(sDelete);
          DeleteStatement.setInt(1,idavviso);
          DeleteStatement.executeUpdate();
          DeleteStatement.close();
          connection.commit();
          return 0;
          }
     catch (SQLException sqle)
        {
           //Errore nell'eseguire la query di cancellazione
           System.err.println("Avviso Exception: Delete: " + sqle);
           return 1;
        }
  }

    public int Insert(int idinsegnamento, String dataAvviso,Strin g
autore,String titoloi,String descrizionei)
    {

     String sInsert;
     sInsert="INSERT INTO
AVVISO(IDAVVISO,IDINSEGNAMENTO,DATA,AUTORE,TITOLOI,DESCRIZIONEI)
VALUES(?,?,?,?,?,?);";


     try {

          PreparedStatement InsertStatement=
connection.prepareStatement(sInsert);

          System.out.println("Avviso: Insert: preparazione della query di
inserimento");
          System.out.println(dataAvviso);
          System.out.println(autore);
          System.out.println(titoloi);
          System.out.println(descrizionei);
          System.out.println("-------------------------");

          InsertStatement.setInt(1,GetFreeID());
          InsertStatement.setInt(2,idinsegname nto);
          InsertStatement.setDate(3,Date.valueOf(dataAvviso));
          InsertStatement.setString(4,autore);
          InsertStatement.setString(5,titoloi);
          InsertStatement.setString(6,descrizionei);

          System.out.println("Avviso: Insert: Esecuzione della query");
          InsertStatement.executeUpdate();
          InsertStatement.close();
          connection.commit();
          System.out.println("Avviso: Insert: Inserimento completato con
successo");
          return 0;
```

```
        catch (SQLException sqle)
        {
         //Errore nell'eseguire la query di inserimento
         System.err.println("Avviso Exception: Insert: " + sqle);
         return 1;
        }

    }


    public int Update( String dataAvviso,String  autore,String titoloi,String
descrizionei, int whereIDAvviso)
    {

     String sUpdate;
     sUpdate="UPDATE AVVISO SET  DATA=? , AUTORE=? ,  TITOLOI=? , DESCRIZIONEI=?
, WHERE IDAVVISO=?;";

     try {

          PreparedStatement UpdateStatement=
connection.prepareStatement(sUpdate);

          System.out.println("Avviso: Update: preparazione della query di
update");
          System.out.println("Data: "+dataAvviso);
          System.out.println("Autore: "+autore);
          System.out.println("TitoloI: "+titoloi);
          System.out.println("DescrizioneI: " + descrizionei);
          System.out.println("-------------------------");



          UpdateStatement.setDate(1,Date.valueOf(dataAvviso));
          UpdateStatement.setString(2,autore);
          UpdateStatement.setString(3,titoloi);
          UpdateStatement.setString(4,descrizionei);
          UpdateStatement.setInt(5,whereIDAvviso );
          System.out.println("Avviso: Update: Esecuzione Query Aggiornamento");
          UpdateStatement.executeUpdate();
          UpdateStatement.close();
          connection.commit();
          System.out.println("Avviso: Update: Aggiorna mento completato con
successo");
          return 0;
         }

        catch (SQLException sqle)
        {
         //Errore nell'eseguire la query di inserimento
         System.err.println("Avviso Exception: Update: " + sqle);
         return 1;
        }

    }


     /**Restituisce un intero che rappresenta il primo valore disponibile di
IDAVVISO<BR>
       Probabilmente necessita di una revisione!!!  */
    public int GetFreeID()
```

```
      Statement tmpStat;
      ResultSet tmpRes;
      int       iFreeID;


      iFreeID=0;

      try {
            tmpStat = connection.createStatement();
            tmpRes=tmpStat.executeQuery("SELECT * FROM AVVISO ORDER BY IDAVVISO
DESC;");

            tmpRes.next();
            iFreeID=tmpRes.getInt("IDAVVISO")+1;
            System.out.println("GetFreeID:  " + iFreeID);
           }
      catch (SQLException ex)
          {
            System.out.println("Avviso: GetFreeID exception: " + ex.toString());
            System.out.println("Avviso: GetFreeID exception: " + iFreeID);
            return 0;
          }
      catch (java.lang.VerifyError ex) //Introdotta per compatibilita Interbase
Inteclient JDBC
          {
            System.out.println("GetFreeID exception : " + ex.toString());
            System.out.println("GetFreeID exception: " + iFreeID);
            return 0;
          }
      return iFreeID;
    }

    public void MoveNext()
    { //Si sposta sulla news successiva
        try {
            resultset.next();
            }
        catch (SQLException ex)
            {

            }
    }

    public void MovePrevious()
    {//Si sposta sulla news precedente
        try {
            resultset.previous();
            }
        catch (SQLException ex)
            {

            }
    }



    public String GetData()
    {//Restituisce una stringa contenete il link
        try{
           Data=resultset.getString("DATA");
           System.out.println("Avviso:GetData:" +Data);
           }
```

```
        {
        System.out.println("Avviso Exception:GetData: " +ex.toString());

        }
        return Data.toString();
    }

    public Date GetObjData()
    {//Restituisce una stringa contenete il link
        try{
           java.sql.Date ObjData=resultset.getDate("DATA");
           System.out.println("Avviso: GetData:" +ObjData.toString());
           return ObjData;
           }
        catch  (SQLExc eption ex)
        {
        System.out.println("Avviso Exception: GetData: " +ex.toString());
        return null;
        }
    }


    public String GetAutore()
    {//Restituisce una stringa contenete il cognome
        try{
               Autore=resultset.getString("AUTORE");
               System.out.println("Avviso: GetAutore:" +Autore);
           }
           catch  (SQLException ex)
          {
            System.out.println("Avviso Exception: GetAutore: " +ex.toString() );
          }
        return Autore.toString();
    }

     public String GetTitoloI()
    {//Restituisce una stringa contenente l'Email del docente
       try{
           TitoloI=resultset.getString("TITOLOI");
           System.out.println("Avviso:  GetTitoloI:" +TitoloI);
          }
        catch  (SQLException ex)
        {
          System.out.println("Avviso Exception: GetTitoloI: " +ex.toString());
        }
        return TitoloI;
    }

    public String GetDescrizioneI()
    {//Restituisce una stringa contenente il Telefono del docente
       try{
           DescrizioneI=resultset.getString("DESCRIZIONEI");
           System.out.println("Avviso: GetDescrizioneI: " +DescrizioneI);
          }
        catch  (SQLException ex)
        {
          System.out.println("Avviso Exception: GetDescrizioneI: "
+ex.toString());
        }
        if (DescrizioneI==null) return "";
        else return DescrizioneI.toString();
    }
```

Servlet: ListDoc.java

```
/* $Id: ListDoc: Lista dei docenti $
 *  Versione 0.5
 */


//user define class
import ListDocLayout.*;
import Docente.*;

//XMLC need class
import org.enhydra.xml.io.*;
import org.enhydra.xml.io.DOMFormatter.*;
import org.enhydra.xml.xmlc.html.*;
import org.enhydra.xml.xmlc.html.HTMLObjectImpl.*;

import org.enhydra.xml.xmlc.XMLCError;
import org.enhydra.xml.xmlc.XMLCUtil.*;
import org.enhydra.xml.xmlc.dom.XMLCDomFactory;
import org.w3c.dom.*;
import org.w3c.dom.html.*;


import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

import java.util.Vector;
import java.sql.*;




/**
 * The simplest possible servlet.
 *
 * @author James Duncan Davidson
 */

public class ListDoc extends HttpServlet {


    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        int j;
        String driverName;
        String url;
        String user;
        String passwd;

        Docente Doc;
        //Layout della pagina della lista dei Docenti file di classe ottenuto
con XMLC
        ListDocLayout DL=new ListDocLayout();
```

```
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        //Impostazione dei parametri per il collegamento al DB
        //Acquisizione dal web container
        url=getServletConfig().getInitParameter("url");
        user=getServletConfig().getInitParameter("user");
        passwd=getServletConfig().getInitParameter("passwd");
        driverName=getServletConfig().getInitParameter("driver");
        // Fine impostazione parametri


        //L'esecuzione della query viene fatta nel costruttore della classe
Docente
        Doc=new Docente(url,driverName,user,pass wd);

        //lementi della tabella da ricopiare interativamente
        HTMLTableElement table = DL.getElementProfTable();
        HTMLTableRowElement sampleRow = DL.getElementResultRow();

       System.out.println("ListDoc: Numero  di Docenti Complessivo: " +
Doc.Count());

       for (j=0; j<Doc.Count();j++)
        {

            Doc.MoveNext();

            System.out.println("Creazione della riga: " + j);
            //HTMLTableRowElement dataRow
=(HTMLTableRowElement)sampleRow.cloneNode( true );

            //Acquisizione dei valori nei campi del record

            try
            {
              //Impostazione del nome e cognome del docente
              Node itemFullname = DL.getElementFullname();
              //Impostazione del HYPERLINK per la home page del docente
              DL.setTextFullname(Doc.GetNome() + " " + Doc.GetCognome());
              HTMLAnchorElement anchorFullName= DL.getElementFull name();
              anchorFullName.setHref("/servlet/DocPage?ID="+Doc.GetID());

              //Impostazione del HYPERLINK per la mail
              DL.setTextEmail(Doc.GetEmail());
              HTMLAnchorElement anchorMail= DL.getElementEmail();
              anchorMail.setHref("mailto:"+Doc.GetEmail());
            }
            catch (NullPointerException E)
            {
                System.out.println("ListDoc:Errore Puntatore a Nullo!");

                //Impostazione dell'hyperlink dell'email in caso di valore nullo
                DL.setTextEmail("");
                HTMLAnchorElement anchor= DL.getElementEmail();
                anchor.setHref("");
            }

            //Clonazione delle varie righe
            HTMLTableRowElement dataRow
=(HTMLTableRowElement)sampleRow.cloneNode( true );

            //Appendere alla tabella
```

```
    }
    //Rimozione della riga servita da modello nel Layout
    table.removeChild(sampleRow);
    //Stampa sul contesto del documento
    out.print(DL.toDocument());

}

}
```

Servlet : DocPage.java

```
/* $Id: LisDoc: Lista dei docenti $
 *  Versione 0.5
 */



//user define class
import ListDocLayout.*;
import DocPageLayout.*;
import Docente.*;
import Insegnamento.*;

//XMLC need class
import org.enhydra.xml.io.*;
import org.enhydra.xml.io.DOMFormatter.*;
import org.enhydra.xml.xmlc.html.*;
import org.enhydra.xml.xmlc.html.HTMLObjectImpl.*;

import org.enhydra.xml.xmlc.XMLCError;
import org.enhydra.xml.xmlc.XMLCUtil.*;
import org.enhydra.xml.xmlc.dom.XMLCDomFactory;
import org.w3c.dom.*;
import org.w3c.dom.html.*;


import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

import java.util.Vector;
import java.sql.*;




/**
 * Servlet per disegnare la pagina dei docenti
 *
 * @author Ciocci Luca
 */

public class DocPage extends HttpServlet {
  String driverName;
```

```
  String url;
  String user;
  String passwd;
  DocPageLayout DPL;

    public void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws IOException, ServletException
    {

        String IDDocente;
        String query;  //Query SQL per la selezione di un docente sulla base del
suo ID
        int j;


        Docente Doc;
        //Layout della pagina della lista dei Docenti file di classe ottenuto
con XMLC
        ListDocLayout LDL=new ListDocLayout();
        DPL=new DocPageLayout();
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        //Impostazione dei parametri per il collegamento al DB
        //Acquisizione dal web container
        url=getServletConfig().getInitParameter("url");
        user=getServletConfig().getInitParameter("user");
        passwd=getServletConfig().getInitParameter("passwd");
        driverName=getServletConfig().getInitParame ter("driver");
        // Fine impostazione parametri



        //Verifico la presenza del parametro ID docente che deve essere passato
dalla
        //classe ListDoc se tale parametro non c'è eseguo un redirect verso la
pagina
        //della lista dei docenti

        IDDocente=request.getParameter("ID");
        if (IDDocente!=null)
            {
              //La ricerca del docente trova solamente un record, in quanto
              //IDDocente è chiave alterna ta
              query="SELECT * FROM DOCENTE WHERE IDDOCENTE="+IDDocente+";";
              Doc=new Docente(url, driverName, user, passwd,query);
              Doc.MoveNext();

              //Riempimento della pagina HTML con gli opportuni  valori del
docente
              DPL.setTextFullname(Doc.GetNome() +" "+Doc.GetCognome());
              DPL.setTextRuolo(Doc.GetRuolo());
              DPL.setTextUfficio(Doc.GetUfficio());
              DPL.setTextTel(Doc.GetTel());
              DPL.setTextFax(Doc.GetFax());
              DPL.setTextRicevimento(Doc.GetRicevimento());
              PrintInsTable(IDDocente);

              //Impostazione del HYPERLINK per la mail
              if (Doc.GetEmail()!=null)
                    {
                      DPL.setTextEmail(Doc.GetEmail());
                      HTMLAnchorElement anchorMail= DPL.getElementEmail();
```

```
                    }
              out.print(DPL.toDocument());
            }
        else response.sendRedirect("/servlet/ListDoc");
 }



  private void  PrintInsTable(String IDDoc)
  {
    Insegnamento ins;  //Oggetto della classe insegnamento
    String sTemp; //Stringa temporanea
      int i;
                //Ciclo per riempire la tabella degli insegnamenti
                ins=new
Insegnamento(url,driverName,user,passwd,Integer.parseInt(IDDoc));

                HTMLTableElement table = DPL.getElementInsTable();
                HTMLTableRowElement sampleRow = DPL.getElementInsRow();
                HTMLTableRowElement dataRow=
(HTMLTableRowElement)sampleRow.cloneNode( true );
                System.out.println("DocPage:PrintInsTable: Numero  insegnamenti "
+ ins.Count());
                for (i=0; i<ins.Count(); i++)
                {
                  ins.MoveNext();

                  //Nome Insegnamento
                  Node itemNome = DPL.getElementInsegnamento();
                  if (ins.GetNome()==null) sTemp="";
                  else sTemp=ins.GetNome();
                  DPL.setTextInsegnamento(sTemp);

                  HTMLAnchorElement anchor= DPL.getElementInsegnamento();
                  anchor.setHref("/servlet/DocInsPage?IDDoc=" +IDDoc + "&IDIns="
+ ins.GetIDInsegnamento());

                  dataRow=(HTMLTableRowElement)sampleRow.cloneNode( true );
                  table.appendChild(dataRow);
             }
            table.removeChild(sampleRow);
    }
}
```

Bibliografia

[1] Marzia Da Como. Progetto del sito web di Facoltà della facoltà di Ingegneria di Modena: parte prima. Tesi di laurea, Università di Modena e Reggio Emilia, Facoltà di In gegneria, corso di laurea in Ingegneria Informatica, 1999 -2000.
http://sparc20.dsi.unimo.it/tesi/index.html

[2] Christian Samuel. Progetto del sito web della Facoltà di Ingegneria di Modena: parte seconda. Tesi di laurea, Università di Modena e Reggio Emilia, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 1999 -2000

[3] Albero Corni e Domenico Beneventano. Progetto sito web Facoltà , 2001

[4] Martin Fowler. UML Distilled. Guida rapida allo Standard Object Modeling Language, Addison Wesley.

[5] Paolo Ciaccia, Dario Maio. Lezioni di Basi di Dati, Progetto Leonardo Bologna

[6] Interbase 6: Getting Started. Installation and Migration, Borland/Intrise.

[7] Interbase 6: Operation Guide, Borland/Inprise

[8] Apache Documentation. Reperibile online all'indirizzo
http://www.apache.org

[9] Apache Server Survivor Guide. Reperibile online all'indirizzo
http://www.bjnet.edu.cn/tech/book/apache/asg01.htm

[10] Tomcat Documentation. Reperibile online all'indirizzo
http://jakarta.apache.org

[11] Enhydra Documentation. Reperibile online all'indirizzo
http://www.enhydra.org

[12] XMLC Documentation. Reperibile online all'indirizzo
http://xmlc.enhydra.org

[13] Bruce Eckel. Thinking in java 2nd Edition Revision 3 reperibile dal sito
http://www.bruceeckel.com

[14] Building Web Components , Forte for Java Internet Edition reperibile dal sito
http://www.sun.com

[15] World Wide Web Consortium. Extensible Markup Language (XML)
http://www.w3c.org

[16] World Wide Web Consortium. Extensible Hypertext Markup Language (XHTML)
http://www.w3c.org