UNIVERSITÀ DEGLI STUDI DI MODENA
E REGGIO EMILIA
Facoltà di Ingegneria – Sede di Modena
Corso di Laurea in Ingegneria Informatica

# Un motore di news per portali Web: progetto ed implementazione

Relatore

Chiar.mo Prof. Sonia Bergamaschi

Tesi di Laurea di

Roberto Delfini

Anno accademico 2000 – 2001

# Sommario

# Indice delle figure

**Indice delle figure**

# Introduzione

In meno di dieci anni la nostra vita è stata letteralmente rivoluzionata da innovazioni in cui l'informatica ha avuto un ruolo preponderante.

Il cambiamento ha investito soprattutto il modo di comunicare ed ha aperto uno scenario inimmaginabile, spazzando via tutte le incertezze che avevano portato a sottovalutare lo sviluppo dell'informatica nel nuovo millennio.

La punta dell'Iceberg di tutto ciò si chiama Internet e non a caso rappresenta il contenitore in cui sono racchiusi ed applicati tutti gli sforzi del mercato per fare in modo che la tecnologia sia alla portata del maggior numero di persone possibile.

Internet non è stata inventata, ma è cresciuta lentamente passando dalle mani dei ricercatori universitari, a quelle di chiunque sentisse l'esigenza di comunicare ed informarsi.

Lo scenario che abbiamo di fronte oggi non è privo di interrogativi e non è immune dai grandi errori di valutazione che sono stati commessi nel passato. La diffusione del Web è tale che è ritenuto indispensabile per qualunque azienda, negozio o privato, essere presente in rete per dare al mondo intero la possibilità di vedere i prodotti, conoscere le idee o semplicemente stabilire un contatto.

Tutto è stato reso possibile dalla maggiore attenzione dedicata alle interfaccie, cioè a quello che permette ad un utente di interagire con un sistema nel modo più facile. Tale attenzione deve crescere di pari passo con la vertiginosa crescita della tecnologia, per evitare che la mancata diffusione delle innovazioni, ne rallenti lo sviluppo.

In linea con quanto detto fino ad ora, il lavoro svolto costituisce uno sforzo per interfacciare utenti alla rete e permettere loro di pubblicare informazioni pur essendo a digiuno di nozioni di HTML e fondamenti di programmazione.

La tesi affronta i problemi di carattere progettuale ed implementativo che sorgono nella realizzazione di un'applicazione server-side che gestisce

l'inserimento di notizie da parte dell'utente in un database e le rende immediatamente disponibili in rete a tutto il mondo.

Tutto ciò prende il nome di 'motore di news' e rappresenta uno dei componenti fondamentali e maggiormente utilizzati di un moderno portale Web il cui primo requisito deve essere il continuo aggiornamento.

Questo tipo di applicazione alleggerisce i webmasters da lavoro poco qualificante e dà la possibilità a chiunque abbia le capacità di usare un browser, di svolgere una funzione fino a ieri riservata a pochi.

Il motore di news rappresenta inoltre un motivo per i visitatori per servirsi del sito, consapevoli che questo è continuamente curato ed arricchito con notizie e servizi sempre nuovi.

Anche nella esperienza maturata in azienda durante lo sviluppo di questa applicazione è emersa l'esigenza di fornire maggiore indipendenza al cliente per quanto riguarda la manutenzione di siti internet di qualsiasi dimensione. La tendenza è quella di fornire prodotti con la possibilità di autogestire l'aggiornamento e di personalizzarlo entro limiti sempre meno stringenti. Nella medesima direzione si muove il nostro operato.

L'idea del motore di News è stata suggerita da un'esigenza di carattere pratico: l'associazione Ex-Alumni dell'università di Modena e Reggio, riteneva necessario arricchire il proprio sito internet della possibilità di pubblicare automaticamente notizie rivolte ai soci.

Interessandoci al problema abbiamo riscontrato un numero elevato di sfaccettature che hanno suggerito di approfondire lo studio di tutto quanto concerne la progettazione e la scelta della tecnologia per un'applicazione di questo tipo ad un livello accademico.

Lo studio culmina con la realizzazione di un prototipo funzionante di un motore di news con le principali funzioni, in grado di essere applicato alla maggior parte dei siti che necessitino di questa tecnologia.

Nel primo capitolo della tesi viene mostrata una breve panoramica sul Web, con un approfondimento delle tecnologie esistenti indicandone le caratteristiche salienti.

Dopo una breve descrizione dei principali Web Server e Application Server in commercio si passano in rassegna i più utilizzati linguaggi di programmazione che hanno trovato applicazione nel Web.

Il secondo capitolo è un'analisi delle problematiche da affrontare per estendere al massimo la portabilità e la personalizzazione di un'applicazione di questo tipo.

Questa analisi non intende fornire una soluzione a tutti questi problemi, ma semplicemente metterli in luce in modo da chiarire i punti di forza e gli eventuali limiti del prodotto.

Con il terzo capitolo si affronta il progetto vero e proprio di un motore di news, facendo uso dell'UML come linguaggio di modellazione.

Questa tecnica permette di addentrarsi progressivamente nel problema, passando da una visione generale ad un dettaglio assai particolareggiato con l'aiuto di schemi e formalismi.

Questa rappresenta la parte più importante del lavoro svolto poiché vengono effettuate le scelte più importanti, partendo dalla definizione degli use cases e dai class diagrams fino ad arrivare ad un passo dall'implementazione.

Per ultimo viene mostrato, per quanto possibile, il prototipo realizzato, giustificando le scelte tecniche ed analizzando i problemi sorti durante la stesura del codice.

In questo capitolo viene illustrato il principio di funzionamento di ogni modulo che compone il sistema riportando stralci di codice in cui sono evidenziati gli aspetti più interessanti.

Infine due appendici completano la trattazione: la prima appendice tratta dell'ambiente di sviluppo utilizzato durante la scrittura del codice, mentre la seconda appendice è un breve manuale d'uso del software per guidare l'utente all'utilizzo delle funzioni implementate.

# Capitolo 1

# Panoramica sul Web

Presentiamo alcuni aspetti introduttivi al mondo del Web, le tecnologie esistenti e i concetti fondamentali fornendo gli strumenti per comprendere le scelte effettuate in fase di progetto.

## 1.1 Server-side, Client-side e HTTP

La prima distinzione che risulta naturale fare, parlando del Web e delle sue applicazioni, è quella fra lato server e lato client. La differenza è tanto netta quanto banale, infatti le applicazioni server-side vengono eseguite dal server, mentre le applicazioni *client-side* girano sui client connessi al sito.

Nella maggior parte dei casi non vi sono dubbi su dove debbano essere eseguiti frammenti di codice, ma vi sono rari casi in cui le moderne tecnologie offrono possibilità di scelta fra le due soluzioni.

Vediamo un po' più in dettaglio cosa accade quando dal browser viene richiesta una pagina generica e come si comporta il server per servire la richiesta.

Il protocollo HTTP (HyperText Transfer Protocol) è stato appositamente creato per il trasferimento di file in formato HTML e si basa su un paradigma di interazione *request/response.*

Tale paradigma si basa su due fasi elementari che sono quella di *request* e quella di *response*. Le due fasi vanno previste nell'ordine corretto; prima il browser invia una *request* e poi il Web fornisce una *response*, formattata nel codice previsto dal protocollo HTTP e la invia al browser. Questo tipo di interazione, prodotta dall'HTTP 1.0, prevede che ad una *request* corrisponda una ed una sola *response* e che la nostra connessione TCP (*Transmission Control Protocol*) venga chiusa al momento dell'arrivo della *response.*

Questo significa che ad ogni richiesta corrisponde una connessione che viene chiusa all'atto della soddisfazione di tale richiesta. Questo comporta che tale protocollo, a differenza di altri, sia *stateless*, cioè non tiene memoria in nessun modo di ciò che è accaduto in richieste passate.

Le applicazioni *server-side* si occupano principalmente di reperire, selezionare e comporre i dati da inviare al client sul quale applicazioni *client-side* li presenteranno nel modo più adatto.

Il codice che gira sul lato client impegna risorse sul computer che effettua la richiesta, questo può divenire utile nei siti molto frequentati in cui l'impegno di risorse nel server diventa un problema. In tali casi è doveroso individuare quelle funzioni che possono essere svolte direttamente da codice interpretato dal browser, come alcuni tipi di controlli sull'inserimento di dati, animazioni grafiche e tutto ciò che può venire in mente per limitare al minimo indispensabile l'impegno di risorse sul server.

L'applicazione del motore di News è interamente *server-side*, poiché deve esaudire le richieste, reperire i dati e costruire pagine da spedire in risposta.

Nel nostro caso l'unico codice eseguito dal client si limita ad effettuare semplici controlli sui dati inseriti dall'utente per verificarne la validità prima di inviarli con la richiesta.

## 1.2 Dinamicità e Staticità

Fino a pochi anni fa, con la diffusione del protocollo HTTP, gran parte delle pagine disponibili in rete erano pagine statiche.

Per pagina statica s'intende un documento che è destinato a rimanere sempre uguale nel tempo a meno che non venga modificato per mano di un programmatore. Il limite di tale staticità è divenuto un problema quando è nata l'esigenza di personalizzare la pagina HTML richiesta dall'utente a seconda delle sue necessità.

Quando si considera l'integrazione tra Web e database, il concetto fondamentale è la *dinamicità*, contrapposto al concetto di *staticità*.

La pagina Web statica riflette, in ogni momento, la situazione esistente nell'istante in cui è stata creata e non quella dell'istante in cui viene letta.

La pagina Web dinamica è una pagina che non esiste a priori, ma viene generata sul momento, a seguito di una richiesta dell'utente. Per questo

motivo rifletterà in ogni momento la situazione esistente nell'istante in cui viene letta.

Affidarsi esclusivamente a pagine di tipo statico, soprattutto in presenza di siti vasti e complessi che richiedono aggiornamenti frequenti, presenterebbe costi di gestione inaccettabili. La modifica delle informazioni contenute nel sito comporterebbe infatti l'aggiornamento manuale di centinaia o migliaia di pagine, richiedendo un grande sforzo per assicurare la coerenza dei contenuti, specie nel caso di singole informazioni richiamate in più pagine.

L'utilizzo di un database come luogo logico dove conservare i dati, da distribuire sul Web tramite pagine dinamiche, rappresenta una radicale soluzione a questi problemi. La robusta e consolidata tecnologia dei DBMS costituisce un ottimo mezzo per gestire grandi quantità di dati in maniera efficiente e flessibile.

Il beneficio principale di questo approccio è una drastica riduzione dei costi di gestione. Un aggiornamento delle informazioni immagazzinate nella base di dati si riflette immediatamente ed automaticamente nei contenuti del sito, senza alcun oneroso intervento di modifica manuale delle pagine. Inoltre, l'utilizzo di un database consente di garantire l'integrità e la coerenza dei dati in maniera semplice e corretta.

Un ulteriore vantaggio è la possibilità di eseguire agevolmente manipolazioni complesse delle informazioni a disposizione, rendendo facile l'introduzione di nuove funzionalità ed aggregazioni di informazioni.

Naturalmente un'applicazione dinamica deve essere progettata in modo adeguato e il risparmio di tempo e manodopera per la sua manutenzione si ripercuote in parte in un aumento del costo dovuto ad una più complessa progettazione e realizzazione.

La necessità di avere a che fare con pagine generate dinamicamente deve essere attentamente valutata in fase di analisi; le variabili in gioco sono molte, ma nella maggior parte dei casi, l'esigenza di dinamicità è chiara fin dalla prima raccolta dei requisiti.

Il primo aspetto da valutare è sicuramente la frequenza con cui devono venire aggiornati i dati. Risulterebbe impensabile dedicare il lavoro di un webmaster alla manutenzione di un sito le cui informazioni cambiano giornalmente, tantomeno quando la figura del redattore, cioè colui che decide i contenuti e quella del webmaster stesso, non coincidono.

Il secondo aspetto da valutare è naturalmente la dimensione, intesa come la quantità di dati che vengono resi disponibili in rete. Mentre risulta accettabile che il sito della piccola attività di famiglia aggiorni

settimanalmente le sue tre o quattro pagine statiche, questo è impensabile per portali di dimensioni maggiori.

Il terzo aspetto chiama in causa il server che deve ospitare il sito. Per la generazione di pagine dinamiche è richiesta una maggiore occupazione delle risorse di memoria e di CPU del server; tale hardware deve essere opportunamente dimensionato per fare fronte al numero di accessi stimato e al numero di pagine visitate.

Inoltre ogni server ha la possibilità di ospitare codice scritto in determinati linguaggi a seconda delle scelte della Server Farm[1].

## 1.3 La tecnologia

Con l'accrescere dell'importanza della rete si è assistito allo sviluppo di un grande numero di prodotti atti a potenziare l'utilizzo del Web sia da parte degli sviluppatori che da quella degli utilizzatori. È quindi doveroso soffermarci sulle tecnologie che hanno fatto del Web ciò che possiamo vedere oggi e che saranno protagoniste del suo sviluppo in un prossimo futuro.

### 1.3.1 Web Server

Sono quelle applicazioni in grado di soddisfare le richieste del Browser rendendo disponibili pagine HTML.

Il prodotto maggiormente utilizzato è Apache, il quale funziona su Windows (95/98/NT), su OS/2 e su tutte le varianti principali di UNIX e basa il suo successo sulla sua affidabilità, trasparenza, modularità, robustezza e sul continuo aggiornamento oltre che sulla distribuzione gratuita. Quest'ultima caratteristica è proprio quella che ha creato le migliori condizioni per lo sviluppo di un software completo in tutti i sensi, usato da più della metà dei siti presenti sul Web. La sicurezza, le prestazioni e la robustezza di Apache sono incontestabili e proprio la distribuzione pubblica del codice sorgente ha fatto si che si sia diffuso rapidamente sottoponendolo ad un esame accurato da parte del pubblico che ha permesso l'individuazione dei problemi di sicurezza e della loro immediata segnalazione. Tutto ciò ha portato ad avere un pacchetto

[1] Azienda che mette a disposizione potenza di calcolo e spazio disco.

estremamente stabile e sicuro che può competere più efficacemente con i pacchetti commerciali sia in termini di velocità che in termini di funzionalità.

Nonostante la sua solidità, Apache non è rivolto a tutti gli utilizzatori. L'installazione e la manutenzione vengono compiute da riga di comando al contrario di altri Web server commerciali che offrono il supporto di un'interfaccia grafica. Tutto ciò è un vantaggio per alcuni sviluppatori, ma si traduce in un maggiore difficoltà di manutenzione, soprattutto per quegli amministratori che sono ad un primo approccio con un Web server.

Il principale concorrente di Apache è Internet Information Server (IIS) della Microsoft.

La filosofia di questo prodotto è in linea con quella che ha contraddistinto la casa madre in questi anni. IIS è dotato, al contrario di Apache, di un'interfaccia grafica che ne permette l'utilizzo da un pubblico meno esperto.

Questa è la caratteristica fondamentale che porta IIS ad essere il principale concorrente di Apache Web Server. L'interfaccia utente facilita le operazioni di installazione e manutenzione.

Come già detto parlando del prodotto precedente, le caratteristiche elencate possono risultare importanti per un utilizzatore occasionale, ma inutili, se non fastidiose per un webmaster più esperto.

L'altro punto di forza di questo software è la diffusione che ha raggiunto in considerazione del fatto che è incluso in Microsoft Windows NT Server.

### 1.3.2 Application Server

Fino ad ora abbiamo parlato di software in grado di soddisfare le richieste di pagine HTML, cioè statiche.

Abbiamo però già spiegato l'importanza rivestita dai siti dinamici ed è doveroso parlare dei prodotti che esaudiscono le richieste dinamiche, cioè gli Application server.

Si comprende fin d'ora che Web server e Application server non possono essere indipendenti fra di loro, ma devono lavorare in modo complementare poiché non è possibile sapere a priori se la pagina richiesta è statica o dinamica.

Solitamente il Web server rimane in attesa delle richieste e soddisfa quelle di sua competenza, inoltrando le altre (quelle dinamiche) all'application server.

Questo meccanismo a cui abbiamo appena accennato, comporta uno sforzo da parte dei produttori di Apache e IIS a adattarsi al prodotto che domina il settore: Tomcat.

La politica commerciale di Tomcat è la stessa di Apache ed è il prodotto leader del mercato con le sue caratteristiche di modularità che ne permettono l'adattamento alle più specifiche tecnologie.

Anche l'IBM con *WebSphere Application server* fornisce un valido prodotto per un utilizzo professionale a qualsiasi livello.

### 1.3.3 Programmare il Web

Affrontiamo ora la parte più interessante della nostra panoramica sul Web, cioè gli strumenti per generare pagine dinamiche.

La dinamicità sul Web diventò importante quando si sentì il bisogno di superare i limiti dell'HTML e si diede origine alla Common Gateway Interface (CGI).

Il CGI è uno standard le cui specifiche definiscono un modo per mezzo del quale il server Web comunica con i programmi esterni e viceversa per far sì che il programma esterno possa generare HTML, immagini, o altro. E' uno standard molto ben definito e utilizzato e senza CGI, produrre materiale dinamico sarebbe stato impossibile.

In breve diventò automatico l'utilizzo di un database per garantire l'integrità dei dati e si aprirono così nuove interessanti prospettive.

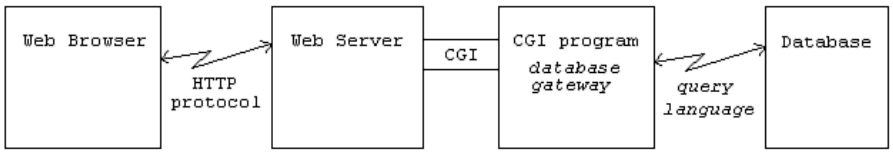


**Figura 1: ruolo dell'interfaccia CGI in un sistema Client-Server**

Questa interfaccia mette in grado i web server di chiamare script per ottenere (o inviare) dati da database, documenti, altri programmi, e presentare questi dati a dispositivi per la visualizzazione tramite il Web. Tuttavia la tecnologia CGI ha un certo numero di limitazioni.

La prima è rappresentata dal fatto che il codice all'interno del CGI script che accede a risorse come il file system o un database, deve essere specifico della piattaforma su cui appoggia il server. Conseguentemente la maggior parte delle applicazioni CGI non funzionano su un server che usa sistema operativo diverso, senza essere modificato. Questo limita la sua utilità in ambienti distribuiti dove l'applicazione Web necessita di girare su molteplici piattaforme.

Il secondo limite è dovuto al fatto che deve essere creato un nuovo processo ogni qualvolta un CGI script viene invocato. I CGI script spesso richiedono parecchie risorse che in questo modo non possono venire adeguatamente dimensionate. Aumentando l'ammontare dell'Hardware, si permetterebbe alle applicazioni CGI di funzionare correttamente, tuttavia il corretto funzionamento rimane limitato entro il confine imposto dall'hardware presente.

Infine, le applicazione CGI sono di difficile manutenzione poiché combinano contenuti e visualizzazione in un solo codice. Di conseguenza sono necessarie due professionalità differenti per compiere manutenzione sui CGI script.

Numerosi rivenditori di Web server hanno migliorato la CGI per il loro specifico prodotto e hanno sviluppato modi migliori di maneggiare CGI, allo scopo di estendere i loro prodotti. Questo ha permesso lo sviluppo di Web-application sofisticate e basate su CGI.

Questa piccola rivoluzione ha permesso di utilizzare non un solo linguaggio di programmazione, ma ha definito un modo di lavorare che può avere come protagonisti molti dei linguaggi esistenti. Il Perl si è distinto fra essi, per la sua disponibilità gratuita e sicuramente per l’estrema portabilità. Il Perl è facile da usare, dispone di funzioni di sicurezza integrate ed è relativamente veloce.

In origine, venne sviluppato per la manipolazione di testi, per generare report e per la manipolazione di files. I progettisti si dedicarono poi alla sicurezza, implementando funzioni in grado di individuare l’origine dei dati poco sicuri e di analizzarne i flussi.

Benché, col passare degli anni, gli sforzi degli sviluppatori abbiano dato vita a nuovi strumenti, più completi e potenti, il Perl rimane uno dei linguaggi maggiormente supportati dai server e quindi più utilizzati, allungando la vita all’interfaccia CGI.

Tuttavia i problemi di base continuano ad esistere: le CGI-applications sono di tipo 'platform-specific', hanno una pessima adattabilità e sono di difficile manutenzione.

Altri strumenti largamente utilizzati sono le Active Server Pages (ASP).

Le ASP non sono altro che pagine che, invece di essere interpretate dal browser, quindi dal lato client, sono interpretate dal server.

Il funzionamento è il seguente: un navigatore di Internet richiede una certa pagina al server tramite il proprio browser, il server recupera la pagina richiesta, interpreta il codice HTML normalmente, in presenza di codice ASP lo elabora e restituisce in formato HTML il risultato. Quando il server termina l'elaborazione della pagina ASP, restituisce l'intero contenuto al browser del richiedente, che lo visualizza a video.

Questa tecnologia nasce alla Microsoft e il suo uso è supportato dalle piattaforme Microsoft. Il Web server come IIS ha integrato il supporto per ASP, cioè lo Scripting Engine. Ciò significa che, in questo caso, IIS funziona come un Application server.

I limiti sono rappresentati dalla portabilità che si limita ai sistemi operativi Microsoft. Tuttavia, almeno in Italia, ASP rimane il mezzo più economico per affrontare il problema della generazione di pagine dinamiche a patto che il server si appoggi su Windows NT/2000 Server. Queste caratteristiche si traducono nell'impossibilità dell'utilizzo sui server Unix delle Active Server Page e in un impiego poco professionale di tale tecnologia, ma questi difetti sono in parte compensati dai brevi tempi di sviluppo e dalle buone prestazioni in termini di velocità.

Le Active Server Pages sono un esempio di codice di un linguaggio di scripting scritto fra le righe di HTML. Questo modo di lavorare che è largamente usato anche con altre tecnologie, comporta un certo comfort durante la fase di sviluppo, ma si porta dietro alcuni problemi di leggibilità e manutenzione che affronteremo in seguito parlando di altre tecnologie che presentano analogie.

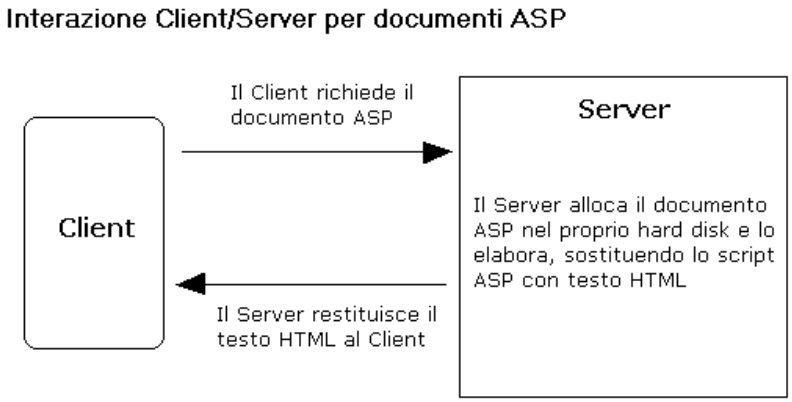


**Figura 2: Interazione Client-Server nella richiesta di una pagina ASP**

La tecnologia relativamente più recente che ha scosso il panorama del Web prende il nome di Java.

L'innovazione sta nell'aver risolto brillantemente il problema della portabilità del codice sorgente e della differenza fra le piattaforme Unix, Mac e Windows.

Java viene utilizzato dal programmatore che vuole creare un programma eseguibile su una qualsiasi macchina grazie alla Java Virtual Machine (JVM) che interpreta il codice (bytecode).

Il bytecode viene generato dal codice sorgente a seguito di un'operazione di compilazione. La JVM è semplicemente un'applicazione che simula una macchina virtuale che è in grado di interpretare questo bytecode. La forza di tutto ciò sta nel fatto che il bytecode generato da un compilatore che lavora su Unix, è uguale a quello generato partendo dallo stesso codice sorgente, ma compilato su Windows NT. Si comprende fin d'ora che questo risultato ha comportato uno sforzo per progettare la JVM e i compilatori per le diverse piattaforme, ma il vantaggio è quello di un'effettiva portabilità totale, imparagonabile con quanto accadeva con il linguaggio C che vantava comunque ottime caratteristiche di compatibilità.

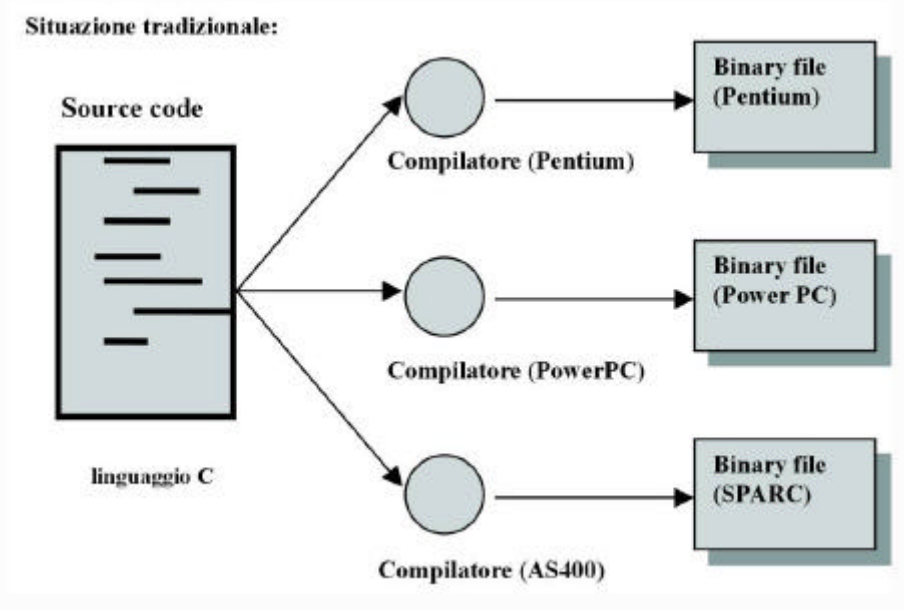


**Figura 3: portabilità del codice eseguibile in linguaggio C**

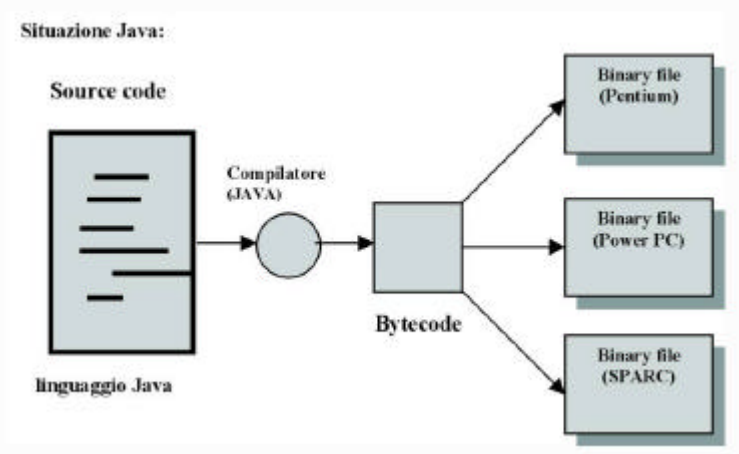


**Figura 4: Portabilità del Java bytecode**

In Java i tipi di dati sono effettivamente gli stessi e il sogno di eseguire lo stesso codice su macchina la cui architettura è radicalmente diversa, si è finalmente avverato.

Dietro tutto questo c'è la Sun, che con il successo riscontrato dalla prima versione di Java, ha arricchito il pacchetto del Java Developement Kit (JDK) con nuove possibilità di sviluppare prodotti destinati al Web. Siamo di fronte ad un linguaggio di programmazione moderno, completo e fortemente Object Oriented. Ma Java non è solo un linguaggio di programmazione: è anche un sistema di runtime, un insieme di strument di sviluppo e un'interfaccia di programmazione di applicazioni (API).

Lo strumento Java maggiormente dedicato al Web sono i Servlet. I Java Servlet rappresentano l'intento di estendere le funzionalità di un Web server. I Servlet possono essere visti come applicazioni che vengono eseguite sul server e sono portabili attraverso differenti piattaforme. Una applicazione basata sull'uso del browser che effettua chiamate ai Java Servlet, non necessita di alcun supporto nel linguaggio di programmazione Java. Questo permette ai Servlet di essere supportati da ogni piattaforma che disponga di una Java Virtual Machine e un Web server predisposto all'uso delle Servlet stesse.

La portabilità attraverso le diverse piattaforme avviene, come per tutto il codice Java, senza la ricompilazione. Inoltre si possono utilizzare API generiche come JDBC per comunicare direttamente con esistenti risorse. Questo semplifica lo sviluppo di applicazioni permettendo alle Web-applications di essere sviluppate più rapidamente.

I Servlet sono estensibili poiché sono basati sul linguaggio di programmazione Java. Questo permette agli sviluppatori l'estensione delle funzionalità di una applicazione per il Web come se fosse una semplice applicazione Java.

I Servlet sono più performanti rispetto ai CGI-script, possono essere caricati in memoria una volta per tutte e richiamati quante volte si vuole, senza che la loro adattabilità richieda hardware aggiuntivo. Una volta che il Servlet è stato caricato in memoria, esso può essere eseguito all'interno di un leggerissimo thread, mentre un CGI-script doveva essere caricato in differenti processi, uno per ogni richiesta.

Questa tecnologia elimina buona parte della complessità in ciò che riguarda la manipolazione dei parametri di una HTTP-request; I componenti hanno diretto accesso ai parametri poiché questi sono rappresentati come oggetti. Con i CGI-script i parametri provenienti da un form HTML, sono convertiti in proprietà di sistema che devono essere lette all'interno del programma.

Uno dei più grossi benefici rimane l'implementazione di un oggetto Sessione visibile da tutta l'applicazione web, in modo che possa interagire con l'utente in modo più semplice.

Il vero strumento per realizzare pagine dinamiche che si appoggia sulla tecnologia Java sono le Java Server Pages (JSP).

La tecnologia JSP è stata progettata per fornire un metodo incentrato sulla presentazione, per sviluppare Servlet. In aggiunta a tutti i vantaggi che i Servlet offrono, le JSP permettono un veloce sviluppo di Servlet i cui contenuti e visualizzazione sono separati. Entrambe le tecnologie Java presentate, descrivono come processare una richiesta (request HTTP) per creare una risposta (response).

Allo stesso modo delle ASP Active Server Pages della Microsoft di cui abbiamo già parlato, le JSP permettono di scrivere codice (in questo caso JAVA) all'interno del codice HTML. Grazie a questo strumento è facile costruire pagine dinamiche con lo stesso linguaggio di programmazione che si è usato per la creazione dei contenuti della pagina. In realtà questa è un'astrazione, nel senso che la pagina, all'atto della compilazione, viene tradotta in un Servlet.

L'importante è che lo sviluppatore non si trovi a dover produrre HTML contenuto fra righe di codice Java, ma viceversa includere righe Java nell'HTML.

Il browser quando richiede la pagina JSP, invoca l'esecuzione di un Servlet che spedisce come risposta un opportuno HTML prodotto dinamicamente.

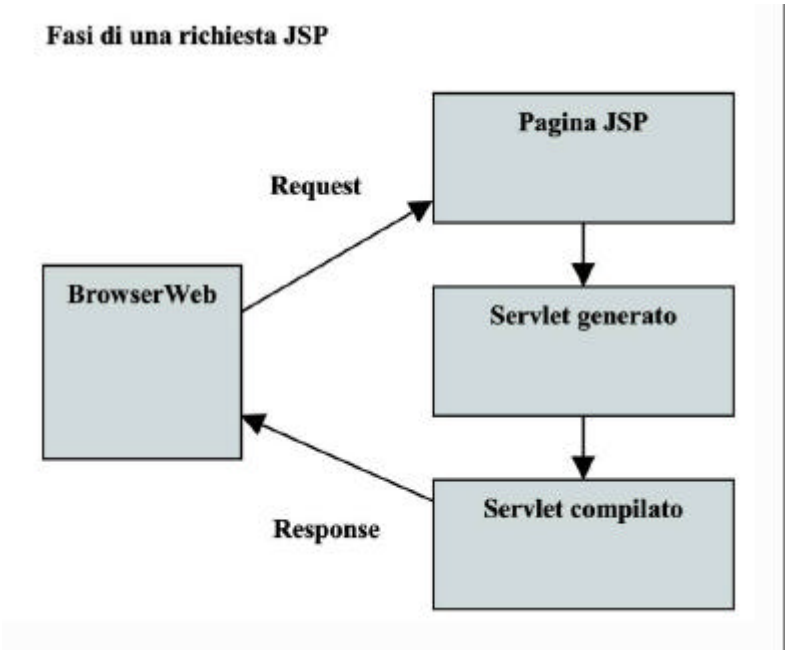


**Figura 5: fasi della richiesta di una pagina JSP**

La convivenza di HTML e Java all'interno di una pagina JSP costringe ad una complicazione dei compiti del grafico, ma solamente per quanto riguarda la leggibilità della pagine che viene peggiorata. Infatti, agendo sull'HTML si è resa realmente indipendente la visualizzazione dal resto.

Tirando le somme, parlando di Java siamo dinanzi ad una tecnologia ricca e completa, in cui viene utilizzato un linguaggio di programmazione sintatticamente uguale al linguaggio C, ma totalmente strutturato ad oggetti, con un numero elevatissimo di classi preconfezionate che lo rendono adatto alla maggior parte delle applicazioni che non richiedono un'elevata velocità di elaborazione, essendo Java un linguaggio interpretato.

### 1.3.4 Connessione alla base di dati

E' chiaro ormai che è necessario attingere i dati da visualizzare da una base di dati che garantisca l'integrità di questi e che gestisca gli accessi concorrenti in scrittura e in lettura in modo rigoroso.

Una volta che abbiamo i mezzi per connetterci al database, siamo in grado di accedere ai dati per mezzo di query di selezione, di aggiornare campi di tabelle e di agire a nostro piacimento sui record.

La tecnologia che prendiamo in esame in questo paragrafo, fa parte del mondo di Java, poiché questo è il linguaggio di programmazione che è stato utilizzato nello sviluppo del motore di news.

L'interfaccia Java Database Connectivity (JDBC) è una API completamente Java, utilizzata per eseguire istruzioni SQL.

JDBC fornisce un insieme di classi e interfacce che possono essere utilizzate dagli sviluppatori per scrivere applicazioni di database. L'interazione più semplice con JDBC può essere divisa in quattro fasi:

1. Apertura di una connessione verso il database.
2. Esecuzione di un'istruzione SQL.
3. Elaborazione dei risultati.
4. Chiusura della connessione verso il database.

JDBC offre il supporto per l'accesso a database il cui modello può essere a due o a tre livelli.

Quando si impiega il modello di accesso a database a due livelli, l'applicazione Java parla direttamente al database. Questo avviene grazie al driver JDBC che invia i comandi direttamente al database. I risultati di questi comandi vengono inviati dal database direttamente all'applicazione.

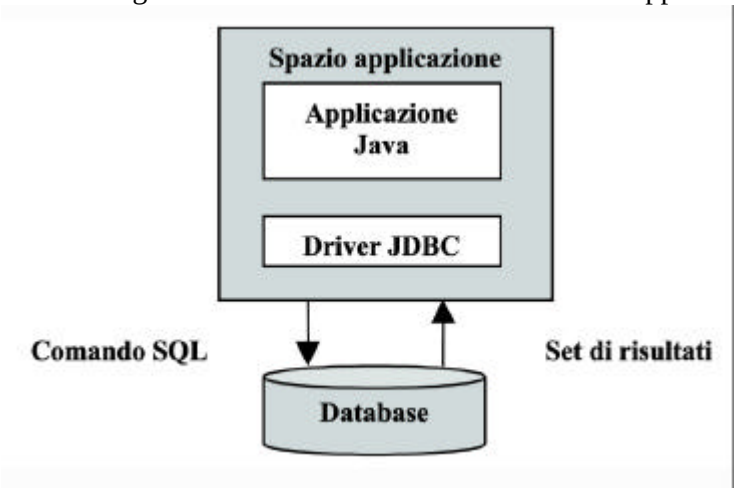


**Figura 6: JDBC a due livelli**

Il modello a tre livelli, come è facile immaginare, è più complicato. Con questo modello di accesso, il driver JDBC invia i comandi ad un livello intermedio, il quale a sua volta li inoltra al database. I risultati di questi comandi sono rispediti al livello intermedio, il quale li restituisce all'applicazione.

La Sun ha definito quattro ulteriori tipologie di driver JDBC.

1. Bridge JDBC-ODBC, più driver ODBC.
2. API nativa, driver parzialmente Java.
3. JDBC-net, driver puro Java.
4. Protocollo nativo, driver puro Java.

Il primo tipo di driver è detto 'ponte', ovvero il bridge JDBC-ODBC. Questo tipo di driver è stato fornito da Sun a partire dalla versione 1.1 del Java Developement Kit (JDK) e consente l'accesso ad un database JDBC attraverso driver ODBC. Il driver ODBC dev'essere già presente e configurato sul client perché il ponte funzioni. Il bridge viene utilizzato praticamente solo per i prototipi o quando non esista alcun driver JDBC disponibile per un particolare DBMS.

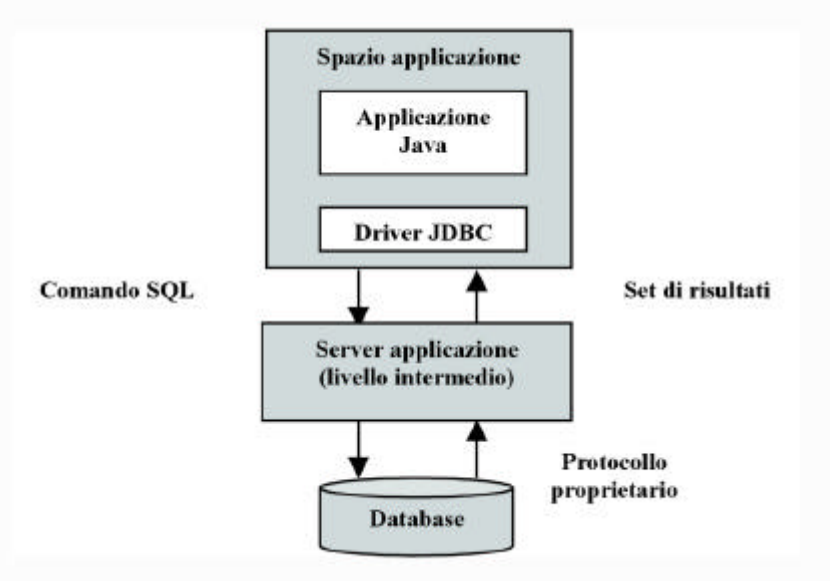


**Figura 7: JDBC a tre livelli**

Il driver API nativa, converte i comandi JDBC in chiamate native specifiche del DBMS. C'è una restrizione analoga a quella dei driver di tipo 1: sul client dev'essere stato preventivamente caricato del codice binario. Questi driver hanno comunque un vantaggio su quelli di tipo 1: si interfacciano direttamente con il database.

I driver JDBC-Net implementano una soluzione a tre livelli. Essi traducono le chiamate JDBC in un protocollo di rete indipendente dal database, compreso da un server intermedio. Questo server traduce il protocollo indipendente dal database, in uno specifico. Le chiamate così modificate vengono inviate al database in questione. Il risultato della chiamata viene reindirizzato al server intermedio e, da questo, al client. Questo tipo di soluzione permette di implementare client puramente Java. Inoltre consente di cambiare database senza modificare in alcun modo i client. Questa è di gran lunga la soluzione JDBC più flessibile.

I driver del quarto tipo, sono in puro Java e comunicano direttamente con i database commerciali. Per fare questo, essi convertono i comandi JDBC direttamente nel protocollo nativo del motore di database. I driver di tipo 4 hanno un vantaggio molto particolare rispetto a tutti gli altri tipi: non richiedono nessuna traduzione o livello aggiuntivo. Conseguentemente, hanno prestazioni molto migliori.

# Capitolo 2

# Problematiche di progetto

I paragrafi che seguono affrontano alcuni aspetti di cui bisogna tenere conto nelle fasi iniziali del progetto. Questo permette di muoversi nella giusta direzione fin dalle prime battute evitando successivamente di trovarsi davanti agli stessi problemi, ingigantiti dallo stato di avanzamento dei lavori e resi per questo insormontabili.

## 2.1 Il pattern strutturale MVC (three tiers)

La maggior parte delle applicazioni monoutente sono comunemente organizzate intorno alle interfacce utente di tipo event-driven.

Lo sviluppatore disegna un'interfaccia utente con uno strumento di progetto, in seguito scrive i blocchi di codice che eseguono le azioni dell'applicazione in risposta ai gesti compiuti dall'utente. Molti ambienti interattivi di sviluppo consigliano questo ordinamento della struttura di codice, dando risalto in primo luogo allo sviluppo dell'interfaccia e ponendosi successivamente le domande riguardo alla funzionalità.

Il risultato è un programma organizzato intorno agli elementi dell'interfaccia utente ed intorno alle azioni dell'utente stesso, in modo che la manipolazione di dati, le funzionalità dell'applicazione e la visualizzazione dei contenuti, risultino completamente intrecciati fra loro.

Questo modo di scrivere codice può essere utilizzato per piccole applicazioni monoutente, in cui le funzionalità richieste cambiano lentamente nel tempo, mentre per applicazioni più complesse è necessario pensare ad un modo diverso di strutturare il lavoro.

Tale riorganizzazione si rende necessaria per una serie di motivi di carattere pratico:

- Le applicazioni più specializzate spesso richiedono una molteplicità di modi di visualizzazione degli stessi dati. L'implementazione del codice per la manipolazione dei dati insieme al codice per la visualizzazione di questi, si traduce in una scrittura ridondante delle stesse procedure che, troppo spesso, è causa di problemi di inconsistenza durante la fase di visualizzazione. Anche se i dati mostrati sono corretti, ogni variazione del modo di visualizzarli richiede l'aggiornamento di codice in punti diversi dell'applicazione, favorendo la presenza di imperfezioni all'interno del software.
- Un'applicazione che richiede una manutenzione a lungo termine, necessita di una struttura che i manutentori possano imparare e capire. Comprendere le funzioni svolte da un'applicazione è ancora più difficile se le fasi di manipolazione logica, formattazione e visualizzazione sono sovrapposte ed intrecciate.
- Lo sviluppo di nuove interfacce utente è reso più oneroso se il codice è sepolto nell'attuale interfaccia utente in coesistenza con altro codice.
- Quando sorge la necessità di arricchire il programma di funzionalità, è difficile reperire il codice sul quale è necessario agire.
- La portabilità dell'applicazione risulta compromessa, soprattutto nel caso di applicazioni (come il motore di news) che si appoggiano ad una base di dati. Tutte le funzioni che richiedono l'esecuzione di SQL, se vengono eseguite in svariati punti dell'applicazione, rendono impossibile o quantomeno estremamente oneroso, ogni sforzo per adattare l'applicazione a database diversi da quello per cui l'applicazione è stata progettata.
- I sistemi distribuiti realizzati in questo modo effettuano spesso accessi ridondanti ai dati, poiché gli elementi dell'interfaccia utente interrogano individualmente il database per informazioni simili.
- La suddivisione in moduli dell'applicazione è più difficoltosa a causa delle dipendenze incrociate delle porzioni di codice.
- Il codice è scarsamente riutilizzabile, poiché ogni componente dipende da molti altri. Questa dipendenza rende il componente ad essere inutilizzabile in altri contesti.

Il disegno di una applicazione distribuita può essere migliorato separando i dati dagli svariati modi in cui essi possono essere raggiunti e

manipolati. Il modello MVC compie una separazione fra i dati dell'applicazione e il modo in cui questi vengono reperiti e visualizzati.

Il pattern MVC si traduce in una separazione fra i tre blocchi funzionali più elementari:

?? Model
?? View
?? Controller

Ognuno dei tre livelli (three tiers) svolge una funzione ben precisa e la loro separazione permette di confinare all'interno di uno stesso ambito tutte le problematiche che hanno un'origine comune.

## *Model*

La parte di Model racchiude i dati e tutte le funzionalità che permettono l'interfacciamento con la base di dati in modo da poter eseguire le istruzioni SQL.

Naturalmente nel caso si debba sostituire il database, sorgono problemi di portabilità della nostra applicazione. In seguito analizzeremo più in dettaglio tutte le possibili problematiche, ma in questa fase dell'analisi ci basta sapere che sono racchiuse entro la parte di Model e solo questa dovrà essere modificata o, nel peggiore dei casi, riscritta.

Inoltre, questa soluzione tiene aperta la possibilità di reperire i dati da qualsiasi altra fonte che non sia un database.

## *View*

Per View si intende tutto ciò che funziona da interfaccia fra l'utente e il sistema. In questa porzione di applicazione sono racchiuse tutte le funzionalità di presentazione dei dati, comprese l'organizzazione dei Menù e la veste grafica. L'importanza del View è chiara pensando al fatto che il Web oltre ad essere contenuti, è esteriorità. Questa è una delle caratteristiche che ha decretato il successo di internet a livello commerciale, permettendo la personalizzazione grafica del Web-site facendo in modo che rispecchi l'immagine dell'azienda.

Dinanzi a queste esigenze è importante che il grafico possa concentrare il suo intervento in una porzione ristretta in cui è di poca importanza il modo in cui vengono reperiti ed elaborati i dati da presentare.

Purtroppo chi cura la veste grafica e l'interfaccia utente, non può essere a digiuno degli aspetti tecnici, soprattutto per quello che riguarda il passaggio dei dati fra client e server. A questo livello di trattazione, la separazione del View è il massimo sforzo a cui possiamo dedicarci, altri dettagli più particolareggiati saranno evidenziati nel corso del progetto.

## *Controller*

Fra View e Model è situato il cuore della nostra applicazione: il Controller. Questa sezione comunica con le precedenti, traducendo le richieste dell'utente in comandi di più basso livello diretti al Model e raccogliendo ed elaborando i dati che quest'ultimo fornisce in risposta, per poi passarli alla View che li renderà visibili nel modo opportuno.

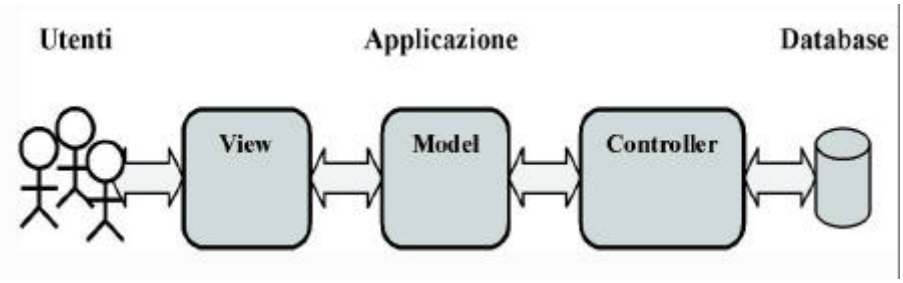


**Figura 8: flussi dei dati in un sistema MVC**

L'utilizzo dello schema MVC permette:

- Lo sviluppo di applicazioni distribuite
- Lo sviluppo di applicazioni di dimensioni notevoli
- Lo sviluppo di applicazioni con un lungo ciclo di vita
- Incremento della portabilità.
- Miglioramento della manutenzione
- Lo sviluppo modulare del software da parte di differenti sviluppatori
- Fase di Test più accurata e più facile

Con questo abbiamo definito uno schema di base che seguiremo, nelle linee principali, durante tutto il progetto e lo sviluppo dell'applicazione.

Infatti lo schema MVC non è da intendersi come un modo di impostare solo l'organizzazione più globale del progetto, ma può essere applicato ricorsivamente all'interno dei singoli livelli della struttura. Questo permette di mantenere la stessa impostazione andando sempre più in dettaglio nel disegno di un'applicazione.

Esistono altri pattern strutturali che affrontano il problema in maniera differente, ad esempio suddividendo il sistema in cinque livelli (five tiers) invece di tre. Senza addentrarci a fondo nella problematica, abbiamo ritenuto necessario introdurre lo schema del pattern a tre livelli poiché rappresenta il primo passo verso l'organizzazione di un'applicazione.

Naturalmente, anche l'implementazione del three tiers model può essere fatta in differenti modi, utilizzando approcci più o meno rigorosi. Sarà compito del software designer valutare di volta in volta la necessità di attenersi rigidamente al modello descritto, ricordando che dal suo utilizzo se ne deve trarre un vantaggio.

Qualora l'utilizzo di un determinato pattern complichi eccessivamente l'impostazione del lavoro senza fornire nessun vantaggio apprezzabile, bisogna abbandonare il pattern utilizzato e cercare un modo più appropriato di affrontare il progetto.

## 2.2 La portabilità

L'aspetto della portabilità è la prima preoccupazione per il progettista di un'applicazione di questo tipo, il cui scopo è quello di essere applicata a siti internet di natura più svariata.

Per portabilità si intende la possibilità di fare funzionare il codice all'interno di ambienti sostanzialmente differenti da quello in cui è stato sviluppato e testato. Tali differenze si possono riscontrare nel sistema operativo, nell'architettura del server, nel database ed infine nel browser utilizzato dal client.

Per questo, fin dalle prime fasi di progetto, bisogna fare molta attenzione impostando il disegno dell'applicazione in modo da non precludere nessuna possibilità in fase di realizzazione. Il pattern strutturale MVC porta sicuramente ad un aumento della portabilità, dovuto alla netta distinzione tra i blocchi funzionali e ad una circoscrizione dei problemi dovuti ad una stessa causa.

Analizziamo ora gli altri aspetti che possono dare origine a problemi di portabilità.

### 2.2.1 Il codice

La scelta delle tecnologie, intese come linguaggi di programmazione, è determinante nella valutazione della portabilità di un progetto.

Abbiamo già passato in rassegna le varie tecnologie e abbiamo messo in luce i loro limiti. Come è stato già anticipato, l'applicazione del motore di news è stata realizzata interamente utilizzando tecnologia Java. Questo per l'ampio panorama in cui Java può essere utilizzato, ma in particolare per la sua portabilità.

Fino a pochi anni fa tutti gli Web-server si affidavano al sistema operativo Unix. Oggi siamo in presenza di una maggiore diversificazione delle macchine, della loro architettura, del sistema operativo con cui lavorano e di conseguenza degli application server. Queste caratteristiche impongono la scelta de Java, il cui bytecode può essere interpretato dalle JVM di macchine con architetture e sistemi operativi sostanzialmente diversi fra loro.

Le sezioni Model e Controller sono state realizzate con tecnologia Java Servlet, mentre il View è interamente realizzato con pagine JSP, funzionanti sia su sistemi operativi Windows 2000 Server (grazie all'interazione fra IIS e Tomcat), sia in ambienti UNIX (utilizzando Apache e Tomcat rispettivamente come Web e Application server). Inoltre l'utilizzo della tecnologia JSP permette l'utilizzo dei Fogli di stile, per la personalizzazione degli aspetti grafici, senza mettere mano al codice dell'applicazione.

Quest'ultima caratteristica permette di compiere un passo avanti nella direzione della tanto ambita separazione fra contenuti e presentazione, a vantaggio della portabilità.

Quindi Java rappresenta, al giorno d'oggi, uno degli strumenti con le migliori caratteristiche di compatibilità; uno dei problemi maggiormente sentiti in un passato recente, è stato brillantemente superato dalla tecnologia. In una realtà in cui per anni i produttori si sono affannati nel definire standard proprietari e a combattere per l'affermazione dell'uno sull'altro, si è rivelata vincente la strategia che porta ad una unificazione di queste realtà separate, a vantaggio di un'auspicata globalizzazione, senza dover subire le situazioni di monopolio che il mercato ha tentato di imporre.

### 2.2.2 Il database

Uno dei punti più vulnerabili ai problemi di portabilità, è quella parte di codice che esegue la connessione al database e l'esecuzione dei comandi SQL.

Purtroppo il mercato mette a disposizione una grande quantità di prodotti ai quali la nostra applicazione deve adattarsi di volta in volta.

Immaginando di dover installare il motore di news in siti dinamici già esistenti, è indispensabile che l'applicazione si integri alla base di dati senza che siano necessarie modifiche del codice.

Come già spiegato all'inizio di questo capitolo, il problema è circoscritto all'interno di un solo modulo, che fa parte di tutto ciò che abbiamo chiamato Model.

Una soluzione al problema della portabilità attraverso diversi tipi di database potrebbe essere quella di scrivere tanti differenti moduli di interfacciamento quanti sono i possibili DBMS con cui presumibilmente potrà lavorare l'applicazione.

Anche se l'impatto di un'operazione di questo tipo interesserebbe una parte ridotta dell'applicazione, sarebbe auspicabile non dover riscrivere nemmeno una riga di codice individuando quali sono le funzionalità che creano differenze e incompatibilità fra i vari tipi di DBMS ed evitando di farne uso, lasciando questa come l'ultima possibilità nel caso che non si riesca a scrivere un unico codice completamente portabile.

Il primo passo verso il nostro obiettivo è l'utilizzo di SQL standard, senza farsi conquistare dagli SQL proprietari che estendono le possibilità di manipolazione dei dati.

Uno dei DBMS che si è rivelato una grande fonte di problemi di portabilità, è Microsoft Access. Questo prodotto è dedicato ad applicazioni di basso livello, sostanzialmente monoutente ma, grazie alla diffusione di Windows e ad una interfaccia utente dotata di funzioni che ne facilitano l'uso a utenti non esperti, ha raggiunto una certa popolarità e per questo deve essere preso in considerazione.

L'utilizzo di tale prodotto in fase di test, permette allo sviluppatore di lavorare velocemente e di localizzare i problemi in maniera piuttosto efficace.

Il problema di Access è che utilizza dei tipi di dato proprietari che non sono presenti nella maggior parte dei DBMS sul mercato, minando la portabilità del sistema.

I tipi di dati che per una applicazione come la nostra possono creare dei problemi sono fondamentalmente tre:

?? Counter
?? Boolean
?? Date

Quando un campo contiene dati di tipo Counter, contiene sostanzialmente un intero lungo che viene automaticamente incrementato di un'unità all'atto dell'inserimento di un nuovo record in tabella, rispetto al valore che assumeva nel record precedente. Questo permette di definire piuttosto comodamente una chiave surrogata autoincrementante, ma limita in maniera inaccettabile la possibilità di manipolare tale dato, rendendo inaccessibile il valore del campo in questione fino al momento in cui si è terminata la fase d'inserimento.

Altri DBMS in commercio hanno il campo autoincrementante con caratteristiche differenti che, nella maggior parte dei casi, creano problemi di compatibilità rendendo difficoltosa l'esecuzione delle query.

Il problema si risolve creando una tabella il cui scopo è quello di contenere un 'numeratore', cioè un campo numerico intero che si deve incrementare di una unità ogni volta che si intende inserire un nuovo record in un'altra tabella che necessiti di un Id numerico autoincrementante. In questo modo deve essere gestito rigorosamente il problema della concorrenza degli accessi nella tabella del numeratore, da cui attingiamo degli interi simulando l'autoincremento.

Il prezzo da pagare è minimo poiché, come nel nostro caso comporta la creazione di una sola tabella aggiuntiva contenente un solo record, composto da un campo chiave e un campo valore. Inoltre una classe avrà il compito di accedere a tale record e incrementare il campo valore.

Il dato di tipo boolean, molto usato in tutti i linguaggi di programmazione, porta a definire alcuni campi dello stesso tipo nelle tabelle di Microsoft Access. Questo limita estremamente la portabilità poiché praticamente nessun altro DBMS permette di definire questo tipo di dato. Inoltre il problema si complica poiché Access implementa il valore true con il l'intero –1 e il valore false con 0.

La soluzione è semplice: non utilizzare mai campi boolean, sostituendoli con interi e gestendo a proprio piacimento il valore assegnato a true e false.

Il terzo tipo di dato che crea problemi è il campo Date. Spesso capita di manipolare un oggetto rappresentante una data, usufruendo delle funzioni offerte dalle API del JDK. Quando si inserisce il valore nel database sorgono problemi di formato dovuti alle molteplici possibilità di rappresentare la data, utilizzate in tutto il mondo e interpretate in modo arbitrario dai singoli DBMS.

Java offre la possibilità di rappresentare una data utilizzando un intero lungo che rappresenta il numero di millisecondi trascorsi dallo scoccare della mezzanotte del 31 Dicembre 1969. Questo numero viene memorizzato nel database come un intero lungo e convertito all'occorrenza in una data effettiva (con possibilità di rappresentare anche l'ora), utilizzando alcuni classi preconfezionate delle API[2].

Questi semplici accorgimenti permettono di rendere il database più portabile attraverso i DBMS in commercio evitando alcuni gravi problemi in fase di installazione dell'applicazione.

### 2.2.3 I JavaScript

Fino ad ora ci siamo occupati della portabilità dell'applicazione intesa come la possibilità di effettuare installazione in differenti server. Questo perché il motore di news lavora sul lato server. Ci occupiamo ora dei problemi di portabilità che si possono riscontrare nel lato client.

*JavaScript* è un potente linguaggio di scripting a oggetti che può essere incorporato direttamente nelle pagine HTML e JSP. Consente di creare applicazioni dinamiche interattive che funzionano completamente all'interno di un Web-browser. La sintassi deriva da altri linguaggi di programmazione come C, C++ e Java, rendendolo di facile apprendimento. Inoltre, poiché si tratta di un linguaggio interpretato, l'ambiente di sviluppo è potente e flessibile.

L'utilizzo dei *JavaScript* nella nostra applicazione è limitato ad alcuni controlli che vengono effettuati sul lato client prima di inviare i dati al motore dell'applicazione sul server. Questo modo di utilizzare il linguaggio prende il nome di *Client-Side JavaScript*.

Il problema della portabilità si riferisce, in questo caso, al lato client, ovvero al browser utilizzato che deve prevedere al suo interno un

[2] Application Program Interface, interfaccia per programmi applicativi e risorse predefinite

interprete *JavaScript* in grado di eseguire il linguaggio di scripting. Purtroppo i due prodotti leader del settore (Internet Explorer e Netscape Navigator) non sono stati in grado di trovare un punto d'incontro e malgrado gli sforzi in atto, non è raro trovare frammenti di codice che creano problemi in uno dei due prodotti. Addirittura già l'HTML può creare questi problemi, ma mentre in questo caso si tratta di problemi di carattere estetico, un malfunzionamento di un JavaScript può mandare in crisi aspetti funzionali di un'applicazione impedendone ogni utilizzo.

Il nome *JavaScript* è di proprietà di Netscape. L'implementazione da parte di Microsoft di questo linguaggio è ufficialmente conosciuta con il nome di *JScript.*

Le versioni di *JScript* sono più o meno compatibili con le equivalenti versioni di *JavaScript*, anche se *JScript* ha saltato una versione passando dalla compatibilità con *JavaScript 1.0* a quella con *JavaScript 1.2*.

*Javascript* è stato standardizzato dall'ECMA (European Computer Manufactures Associations) e da parte dell'ISO (International Standards Organization). Gli standard sono l'ECMA-262 e l'ISO-16262. Questi standard definiscono un linguaggio ufficialmente noto come ECMAScript, approssimativamente equivalente a *JavaScript*, anche se non tutte le implementazioni *JavaScript* sono conformi a tutti i dettagli dello standard ECMA. In particolare, soltanto *JavaScript 1.3* può dirsi pienamente compatibile con lo standard ECMA-262.

Le diverse versioni dei browser incorporano versioni differenti degli interpreti *JavaScript*. In tabella sono mostrate le versioni di base supportate dai due browser più diffusi.

| Versione del browser | | Microsoft Internet Explorer |
|---|---|---|
| 2 | JavaScript 1.0 | |
| 3 | JavaScript 1.1 | JavaScript 1.0 |
| 4 | JavaScript 1.2; non pienamente compatibile con lo standard ECMA-262 | JavaScript 1.2; conforme con lo standard ECMA-262 |

**Figura 9: Compatibilità fra Browser e JavaScript**

Fortunatamente esiste ancora un notevole insieme di caratteristiche di *JavaScript* lato client su cui entrambi i browser concordano e, per il modesto uso che è stato effettuato in questo progetto del linguaggio *JavaScript*, non si dovrebbero riscontrare problemi di portabilità, se non con versioni piuttosto datate dei browser.

## 2.3 La Personalizzazione

Lo scopo dell'applicazione è quello di fornire la possibilità di effettuare la pubblicazione in rete di notizie con caratteristiche predefinite, senza fare uso di HTML, ma semplicemente interagendo con un'interfaccia utente.

Questo assume importanza qualora rappresenti una funzione con cui arricchire un portale Web, per incentivarne la frequentazione da parte dei visitatori. È interessante prevedere la possibilità di agire su alcuni aspetti dell'applicazione in modo che il webmaster, o addirittura l'utente finale, possa personalizzare secondo il proprio gusto l'interfaccia utente e le funzionalità dell'applicazione.

### 2.3.1 La veste grafica

L'utilizzo del pattern strutturale MVC ha permesso di racchiudere entro la sezione View tutto il codice responsabile della visualizzazione dei dati.

Inoltre la tecnologia JSP permette una migliore gestione dell'HTML dinamico di quanto avviene con l'utilizzo di tecnologie meno recenti, come ad esempio l'interfaccia CGI.

Volendo adattare la grafica del motore di news ad esigenze specifiche, si rende necessaria una sostanziale riscrittura delle pagine JSP, solamente per quanto riguarda il codice HTML in esse contenuto.

In figura è rappresentato un esempio di pagina JSP che, in questo caso, svolge la funzione di visualizzare l'elenco delle news che possono essere lette. In neretto è indicato il codice HTML su cui bisogna agire per modificare la grafica della pagina.

```
<%@ page errorPage="errorpage.jsp" %>
<%@ page language ="java" %>
<%@ page import ="news.DBmanager" %>
<%@ page import ="java.util.Enumeration" %>
<%@ page import ="java.util.Vector" %>
<%@ page import ="news.News" %>
<%@ page import ="news.Link" %>
<%@ page import ="news.DateManager" %>
<%@ page import ="java.util.Date" %>
<%@ page import ="news.NewsBean" %>
<html>
<head>
<title>Lettura News</title>
</head>

 <body>
 <font face="verdana">
 <%
       int num_car = 20;
       String creatore;
       String titolo;
       String corpo;
       String corpo_ridotto;
       int IDnews;
       String firma;
       Enumeration enum = (Enumeration)request.getAttribute("EnumBean");
       String link;
       String descrizione;
       int id;
       int tipo;
       int flag=0;
 %>
               <H2 align="center" >ELENCO NEWS</H2>
               <table align ="center" border ="0" width ="600">
 <%
       while (enum.hasMoreElements()){
               NewsBean nb = (NewsBean)enum.nextElement();
               News n = nb.getNews();
               creatore = n.getCreatore();
               titolo = n.getTitolo();
               corpo = n.getCorpo();
               IDnews = n.getIdNews();
               firma = n.getFirma();
               long data = n.getData();
               tipo = n.getTipo();
               Date data_obj = new Date(data);
               String data_str = DateManager.haveDate(data_obj);
               if (corpo.length() > num_car  ){
                       corpo_ridotto =
corpo.substring(0,num_car).concat("...");
               }
               else {
                       corpo_ridotto = corpo;
               }
 %>
                <tr>
                 <td bgcolor="#E6E6FA">
                  <b>Scritta da: </b><%=firma %><b> Data: </b><font
```

```
face="arial"><%=data_str%>
                 </td>
                 </tr>
                 <tr>
                 <td bgcolor="#E6E6FA">
                <a
href="/servlet/news.NewsManager?function_news=espandi&news_da_espandere=<%
=IDnews%>"> <b>Titolo: </b><%=titolo %> </a>
                 </td>
<%
                      Vector vlink = nb.getVlink();
                      if (vlink.size()>0){
                             flag = 1;
                      }
                      if (flag == 1){
%>
                 <td>
<img src="../delfo/News/images/link.gif" align="left" alt="news con link"
border="0">
                  </td>
<%
                             flag=0;
                      }
                      if (tipo > 0){
%>
               <td>
                 <img src="../delfo/News/images/immagine.gif" align="left"
alt="News con immagine" border="0" align="left">
                </td>
<%
                      }
 %>
        </tr>
               <tr>
                 <td>
                  <pre><font face="verdana"><b>Testo: </b><%=corpo_ridotto
%><br></font></pre>
                 </td>
                 <td>
<%
       }
 %>
                </td>
               </tr>
               </table>
 <p align="center"><a href="/delfo/News/home.jsp">[Back]</a>
 </p>
 </font>

 </body>

</html>
```

**Figura 10: codice di un documento JSP**

È facile rendersi conto che nonostante la tecnologia JSP abbia dato la possibilità di inserire codice Java fra le righe di HTML, la leggibilità della pagina risulta compromessa e le modifiche richiedono una certa

dimestichezza con le Java Servlet Page. Per questo motivo ogni riscrittura del codice delle pagine JSP è un'operazione impegnativa che solo chi è a conoscenza di tale tecnologia può effettuare.

L'alternativa è prevedere già in fase di progetto, la possibilità di modificare alcune variabili che agiscono sulle modalità di visualizzazione.

Analizziamo ora i principali metodi per adattare l'applicazione alle esigenze del webmaster e, in alcuni casi, dell'utente finale.

### *Personalizzazioni rivolte al singolo utente*

Attraverso la definizione di variabili private di proprietà di ogni utente registrato, è possibile prevedere alcuni settaggi predefiniti come: il colore dei campi delle tabelle, il tipo, il colore e la dimensione caratteri, il caricamento di un'immagine da utilizzare come sfondo della pagina o alcuni elementi grafici come le icone.

Dal Software sarebbe possibile accedere ad un menù che permetta la modifica di tali caratteristiche. In questo modo ogni navigatore provvisto di username e password per l'accesso al sistema, si troverebbe a lavorare in un ambiente parzialmente personalizzato. Ciò è reso possibile dall'utilizzo del database per salvare le impostazioni che l'utente ha scelto. Tali caratteristiche devono essere previste già in fase di progetto dello schema relazionale del database e devono essere attentamente valutate in base alla mole dei dati che si prevede.

Questo modo di lavorare sta riscontrando successo in alcuni dei portali mondiali più famosi, ma la sua implementazione non è indolore e si traduce in tempi di progettazione più lunghi.

## *StyleSheet*

Attraverso l'uso dei fogli di stile (StyleSheet) è possibile effettuare la scelta di parametri come: i colori, il font del carattere utilizzato, la formattazione del testo, l'aspetto delle tabelle e l'evento collegato alla posizione del mouse.

A livello mondiale è consigliato l'utilizzo degli StyleSheet e la W3C[3] ha addirittura deprecato l'uso di alcuni fra i più usati attributi dei tag HTML in previsione di vietarli nelle future versioni. La situazione reale è sostanzialmente differente, infatti la presenza di molti client sui quali sono installati bowser piuttosto datati, costringe ad un uso ridotto dei fogli di stile a vantaggio della visibilità delle pagine.

Prima di fare uso di tale tecnologia, è necessario effettuare test di visualizzazione, utilizzando differenti browsers nelle loro diverse versioni e valutare la possibilità che la pagina appaia diversa dalle aspettative del Web Designer. I maggiori problemi si riscontrano nell'utilizzo di Netscape Navigator nelle versioni più datate.

## *Il dimensionamento delle immagini*

Prevedendo delle variabili modificabili direttamente dall'interfaccia utente è possibile decidere la dimensione in pixel in cui vengono visualizzate le immagini allegate alle news.

A tale proposito è bene distinguere fra due tipi di ridimensionamento:

?? Ridimensionamento in visualizzazione
?? Ridimensionamento in dimensione

Infatti ridimensionare l'immagine in fase di visualizzazione comporta un adattamento, da parte del browser, dell'immagine allo spazio che le è stato riservato all'interno della pagine. Nel caso in cui si debbano visualizzare immagini di grandi dimensioni in piccoli spazi, è necessario effettuare un reale ridimensionamento che agisca sullo spazio che occupa l'immagine sul disco fisso, questo per evitare un inutile spreco di risorse hardware e limitare il tempo di download del file.

[3] World Wide Web Consortium, ovvero l'organizzazione che a livello mondiale si occupa della standardizzazione dell'HTML

### *L'impostazione della pagina*

Parlare di personalizzazione quando si parla di impostazione della pagina, comporta problemi maggiori. Una facile ed efficace soluzione è quella di prevedere alcuni diversi template rappresentati da differenti pagine JSP.

A seconda delle esigenze verrebbero inviati i dati a una pagina invece che ad un'altra. Questo permetterebbe di modificare radicalmente la grafica dell'applicazione, sempre restando confinati entro le scelte iniziali. Naturalmente questa scelta comporta un aumento del numero delle pagine e un proporzionale incremento dei costi di manutenzione.

### *Configurazione tramite file di testo*

Come avviene in alcuni programmi, esiste la possibilità di configurare gli elementi grafici dell'applicazione tramite l'uso di un file di testo. Naturalmente la scelta degli elementi da modificare e il loro aspetto si ripercuote (come con l'uso dei fogli di stile) su ciò che vede l'intera pluralità di utenti che accedono al sistema.

Anche se ad alcuni lettori può apparire inutile complicare il progetto di un'applicazione dinamica per futili motivi come il colore del testo, ricordiamo che il successo di un moderno portale è fortemente legato, oltre che ai contenuti, all'aspetto esteriore, il cui giudizio rappresenta una valutazione troppo personale, e la personalizzazione risolve il problema nella maggior parte dei casi.

## 2.3.2 Le funzioni

Personalizzare le funzioni offerte all'utente di un sistema è un'operazione non priva di ostacoli.

Discutiamo le varie possibilità che si presentano ad un progettista di un software di questo tipo, valutando di volta in volta i limiti della scelta effettuata.

### *Menù con funzioni nascoste*

Il modo più semplice per effettuare una blanda personalizzazione dell'interfaccia, è quello di suddividere gli utenti in gruppi che hanno accesso o meno a determinate funzioni. In base ai diritti di cui è in possesso, ogni utente, interagisce con un menù più o meno ricco di voci, vedendo solo le funzioni a cui è abilitato e ignorando l'esistenza di quelle verso le quali sarebbe interdetto.

### *Macrofunzioni e Microfunzioni*

In riferimento al pattern MVC, è possibile organizzare il tier Controller in modo che svolga delle microfunzioni elementari le quali, combinate assieme, compongono le macrofunzioni selezionabili dall'interfaccia utente. Così ci sarebbe la possibilità di ottenere nuove macrofunzioni combinando opportunamente le microfunzioni disponibili. Il prezzo di un'operazione di questo tipo è una parziale riscrittura del Controller. La sua applicabilità è spesso ristretta ad una aggiunta di funzionalità per quello che riguarda le visualizzazioni dei dati, intese come viste differenti effettuate sulle tabelle. Tuttavia non è escluso che un'impostazione lungimirante in fase di disegno del tier Controller, possa portare ad avere un margine di movimento più ampio.

### *Modularità del Controller*

Facendo riferimento all'utilizzo del pattern strutturale MVC, è interessante impostare il progetto dell'applicazione in modo da lasciare spazio all'aggiunta di funzioni. Siccome il tier interessato è il Controller, la sua modularità è una caratteristica necessaria per la possibilità di aggiungere funzioni vere e proprie. Un'operazione di questo tipo è una prerogativa degli sviluppatori che conoscono bene l'applicazione sulla quale stanno agendo e comporta spesso un impegno non sottovalutabile in termini di tempo.

Dare la possibilità all'utente di creare nuove funzioni direttamente dall'interfaccia, è affascinante, ma la sua realizzazione rimane, a nostro parere, molto difficoltosa, se non impossibile nella maggior parte dei casi.

# Capitolo 3

# La modellazione del software in UML

Per modellazione si intende la costruzione di un modello in grado di descrivere gli aspetti più importanti del problema. Tale strumento è prezioso per lo sviluppatore che si trova impegnato nella stesura finale del codice.

Per effettuare ciò abbiamo scelto uno strumento moderno che rappresenta, in un certo senso, il riassunto di due decenni di sforzi finalizzati a facilitare la progettazione di software.

## 3.1 Introduzione all'UML

Prima di affrontare il progetto vero e proprio dell'applicazione server-side per la pubblicazione automatica di notizie sul Web (motore di news), è doveroso introdurre alcuni strumenti di cui abbiamo fatto uso nella fase di modellazione.

I paragrafi che seguono non vogliono essere una guida esaustiva all'uso del linguaggio di modellazione UML. Il loro intento è quello di guidare il lettore verso le motivazioni che ci hanno portato alla scelta di tale strumento, evidenziandone le peculiarità e i limiti.

L'Unified Modeling Language (UML) è il successore di una serie di metodologie di analisi e progettazione orientate ad oggetti (OOA&D)[4] apparse tra fine degli anni '80 e i primi anni '90. La definizione più diretta sarebbe quella che lo indica come l'unificazione dei metodi di Booch, Rumbaugh e Jacobson, ma la sua portata va oltre.

La diffusione di tali metodi si deve all'affermazione dei linguaggi di programmazione orientati ad oggetti ed all'esigenza, da parte di

[4] Acronimo di Object-Oriented Analysis and Design.

sviluppatori e progettisti, di disporre di strumenti per descrivere in maniera chiara e standardizzata, le interazioni tra questi oggetti.

Si è fatta strada l'esigenza di unificare questi strumenti per fornire un valido punto fermo che permetta l'esposizione di progetti risolvendo molti problemi di comunicazione.

L'UML è stato standardizzato dall'OMG, Object Management Group ed è riconosciuto come uno standard OMG.

L'UML è un linguaggio di modellazione, non un metodo. Gran parte dei metodi consistono sia in un linguaggio di modellazione che in un processo. Un linguaggio di modellazione è la notazione (principalmente grafica) che i metodi usano per esprimere i progetti. Il processo è l'indicazione dei passi da compiere per intraprendere un progetto.

La maggior parte delle volte che si usa un metodo, si utilizza il linguaggio di modellazione, ma raramente si utilizza il processo. In ogni modo il linguaggio di modellazione è la parte più importante di un metodo, svolgendo un ruolo chiave nella comunicazione. Infatti, nella discussione di un progetto, è il linguaggio di modellazione che deve essere conosciuto e capito dalle parti in causa e non il processo utilizzato per affrontare il progetto.

Quello che è stato fatto per i linguaggi di modellazione sotto il nome di UML, è stato fatto anche per i processi e prende il nome di RUP[5]. Tuttavia in questa sede ci disinteressiamo del RUP concentrando la nostra attenzione sull'UML.

### 3.1.2 L'importanza dell'analisi e della progettazione

Lo scopo dello sviluppo del software è la produzione di codice. I diagrammi rappresentano solamente figure gradevoli, ma la loro utilità non è sempre chiara. Nessun cliente ringrazierà una software house per avergli fornito diagrammi accurati, quello che interessa è un software funzionante.

Così, prendendo in considerazione l'UML, è importante chiedersi in che modo può aiutarci nel momento in cui bisognerà scrivere il codice.

La ragione fondamentale è la comunicazione: si usa UML per comunicare concetti in modo più chiaro di quanto si potrebbe fare altrimenti. Il linguaggio naturale è troppo impreciso e quando i concetti si fanno complessi, è insufficiente. Il codice è troppo preciso e distoglie

[5] Rational Unified Process.

l'attenzione dal concetto fornendo una descrizione troppo dettagliata. Lo strumento più adatto deve vantare le caratteristiche di precisione richieste, senza però addentrarsi minuziosamente nei dettagli. La documentazione UML rappresenta un modello, sul quale ragionare, fare correzioni e ipotesi. Il progettista usa l'UML come il viaggiatore usa la cartina geografica. Una mappa in scala uno a uno è inutile e, allo stesso modo, troppi dettagli nel modello lo rendono inefficace nel compiere una sintesi del problema.

Molto spesso il linguaggio utilizzato dal cliente non coincide con il linguaggio dell'analista, creando problemi nella fase iniziale, che troppo spesso vengono alla luce quando il progetto è in uno stadio troppo avanzato.

Con i suoi numerosi diagrammi, l'UML crea un punto d'incontro fra sviluppatore ed esperto di dominio, in modo da indirizzare lo sviluppatore verso la giusta direzione fin dalle prime fasi.

Quando si dispone di una buona documentazione che modella le caratteristiche più rappresentative del mondo reale, si può facilmente passare alla fase di stesura del codice senza incontrare quelle ambiguità funzionali caratteristiche delle analisi non corrette.

### 3.1.3 Use cases (casi d'uso)

Per molto tempo, sia nello sviluppo object-oriented che in quello tradizionale, sono stati utilizzati esempi tipici di interazione per favorire la comprensione delle specifiche. Questi scenari venivano trattati in modo informale e quasi mai diventavano parte integrante della documentazione di progetto.

Per descrivere l'utilizzo degli *use cases* introduciamo il concetto di *scenario*.

Uno *scenario* è una sequenza di passi che descrivono l'interazione fra l'utente e il sistema, ponendo il lettore dinanzi ad una situazione simile a quella incontrata durante l'uso dell'interfaccia utente del software.

Sono necessari scenari separati nel caso in cui il sistema fornisca risposte diverse o nel caso in cui l'utente selezioni funzioni differenti. Lo *scenario* si limita a descrivere a parole una situazione e la sua flessibilità permette di avere una prima superficiale descrizione del sistema.

Un *caso d'uso* è un insieme di scenari legati da un obiettivo comune per l'utente. Ci sono molte variazioni sullo stile che potete adottare per

descrivere il contenuto del caso d'uso; l'UML non indica specificatamente uno standard. Inoltre potrebbero venire aggiunte sezioni addizionali, come le precondizioni che devono essere verificate per trovarsi ad interagire con un certo scenario. Non è necessario includere tutti i dettagli, ma è indispensabile arricchire uno *scenario* con gli elementi che si ritenga che siano i più significativi.

Fino ad ora abbiamo descritto una situazione utilizzando le parole, ma lo strumento più eloquente rimane lo *use case diagram.*

I diagrammi di casi d'uso sono rappresentazioni grafiche che permettono di fornire una visione più generale degli *use cases.*

Un ruolo interpretato dall'utente nei confronti del sistema è sintetizzato con l'uso dell'attore. Nel diagramma di caso d'uso avremo tanti attori diversi quanti sono i possibili ruoli dell'utente o addirittura di parti del sistema. Un utente può interpretare anche più di un ruolo e il compito del diagramma è proprio quello di mettere in luce tali aspetti.

Gli attori interpretano casi d'uso. Un singolo attore può eseguire più casi d'uso; viceversa un caso d'uso potrebbe essere eseguito da differenti attori. In pratica il concetto di attore è utile nella definizione dei casi d'uso stessi, infatti in sistemi molto grandi, l'operazione di suddivisione degli use cases può risultare difficoltosa. In tale situazione, la compilazione di una lista degli attori aiuta nella stesura di una lista dettagliata dei casi d'uso che possono essere interpretati da tali attori.

Benché nel diagramma siano rappresentati come omini stilizzati, gli attori non devono necessariamente essere persone fisiche. Un attore potrebbe essere anche un sistema esterno che cerca di ottenere informazioni dal sistema locale o un modulo diverso della stessa applicazione.

Oltre ai collegamenti fra attori e casi d'uso, si possono mostrare relazioni dirette fra casi d'uso diversi.

La relazione di *inclusione* si verifica quando un determinato comportamento si ripete in più casi d'uso e non si vuole ripetere la sua descrizione.

La *generalizzazione* può essere d'aiuto quando uno use case ha aspetti in comune con un altro, ma lo estende con funzioni aggiuntive.

Spesso l'utilizzo di tali tecniche permette una descrizione più lontana dalla realtà, ma più vicina alla suddivisione funzionale della quale bisogna tenere conto durante la stesura del codice.

### 3.1.4 Class diagrams (diagrammi delle classi)

I *diagrammi delle classi* sono divenuti una tecnica di modellazione object-oriented davvero centrale: in pratica non esiste nemmeno una metodologia che non ne abbia inclusa una qualche variazione.

Essi contengono il maggior numero di concetti utili alla modellazione. Un *class diagram* descrive il tipo degli oggetti che compongono il sistema e le relazioni statiche esistenti tra loro. Ci sono due tipi fondamentali di relazioni statiche:

?? Associazioni
?? Sottotipi

Le prime si riferiscono ai diversi modi in cui si possono relazionare oggetti di classi sostanzialmente diverse, mentre i secondi mostrano legami tra oggetti che hanno molte caratteristiche in comune.

Nella pratica un'associazione è una relazione tra istanze di oggetti. Ogni associazione ha due capi, ognuno dei quali è connesso ad una delle classi dell'associazione. Un capo può venire espressamente etichettato da un termine che prende il nome di *ruolo* dell'associazione. Il capo ha anche un'etichetta chiamata *molteplicità* che rappresenta il numero di istanze che possono prendere parte alla relazione. La molteplicità è solitamente composta da due numeri che indicano l'estremo inferiore e superiore del numero degli oggetti.

Quando su un'associazione è indicata una freccia, questa indica la *navigabilità*, cioè il senso nel quale bisogna leggere la porzione di diagramma. Nel caso in cui non sia indicata la navigabilità, l'associazione è bidirezionale.

Il problema che può sorgere utilizzando i diagrammi delle classi è che sono così ricchi da rischiare di travolgere l'utente con la loro complessità. Per evitare ciò è necessario ricordare che tali strumenti devono sempre semplificare il nostro lavoro e, nel caso si riscontri un aumento della complessità del problema, il loro uso deve essere limitato allo stretto necessario.

Dopo aver descritto sommariamente le caratteristiche di questi diagrammi, è importante chiarire una sottigliezza riguardante il loro uso da parte degli utenti. Normalmente non viene documentata, ma ha un forte impatto sul modo corretto di interpretare un diagramma, dato che definisce implicitamente che cosa si sta effettivamente descrivendo con un modello.

Disegnando un diagramma delle classi si possono adottare tre diversi punti di vista o prospettive:

- **Concettuale.** Se si adotta questo punto di vista, il diagramma rappresenta i concetti propri del dominio che si sta studiando. Questi concetti poi saranno in relazione naturale con le classi destinate ad implementarli, ma spesso non esiste una corrispondenza diretta. In realtà, un modello concettuale, dovrebbe essere disegnato con poco o nessun riguardo verso il software destinato ad implementarlo, cosicché può essere considerato indipendente dal linguaggio. Questo punto di vista è detto *concettuale* o *essenziale*.
- **Specifica.** Ora la prospettiva riguarda il software, ma a livello di interfaccia e non di implementazione. Lo sviluppo object-oriented pone una grande enfasi sulla differenza fra interfaccia ed implementazione, ma nella pratica questa viene spesso disattesa perché, in un linguaggio di programmazione orientato ad oggetti, la nozione di classe combina entrambi gli aspetti. Questo è un vero peccato poiché la chiave per programmare OO[6] in modo efficace è fare sempre riferimento all'interfaccia di una classe, non alla sua implementazione.
- **Implementazione.** Utilizzando questo punto di vista si fa esplicitamente riferimento alle classi e si schematizza l'implementazione vera e propria. Probabilmente è il punto di vista usato più frequentemente, ma spesso è preferibile sotto molti aspetti, adottare una prospettiva a livello di specifica.

Capire la differenza fra i vari punti di vista è cruciale per la realizzazione e la comprensione dei diagrammi delle classi. Sfortunatamente, i confini tra prospettive differenti non sono ben definiti e la maggior parte dei progettisti non si preoccupa di questo aspetto quando comincia a disegnare i diagrammi.

Nella nostra esperienza abbiamo notato che questa confusione non è rilevante se si mescolano diagrammi concettuali e di specifica, ma è fondamentale effettuare una separazione netta fra livello di specifica e quello di implementazione.

[6] Object-Oriented

### 3.1.5 Interactions diagrams (diagrammi delle interazioni)

I *diagrammi di interazione* sono modelli che descrivono come dei gruppi di oggetti possono collaborare nello svolgimento di un'attività.

Tipicamente, un *diagramma di interazione* illustra il comportamento di un singolo caso d'uso. Nel diagramma è mostrato un certo insieme di oggetti e i messaggi che vengono scambiati all'interno di questo insieme durante lo svolgimento dell'use case.

Ci sono due tipi di diagrammi di interazione: i *diagrammi di sequenza* e quelli di *collaborazione*.

All'interno di un *diagramma di sequenza* ogni oggetto viene mostrato come un *box* posto in cima ad una lunga linea verticale tratteggiata. Questa linea è chiamata *linea di vita dell'oggetto* e rappresenta la durata della vita dell'oggetto durante l'interazione in esame.

Ogni messaggio è rappresentato da una freccia posta tra le linee di vita di due oggetti. L'ordine temporale nel quale vengono scambiati i messaggi è mostrato leggendo la pagina dall'alto in basso. È possibile mostrare anche una *self-call* ovvero una chiamata interna di un oggetto ad una delle sue funzioni.

L'uso di condizioni, permette di stabilire quando un messaggio viene effettivamente generato. Infatti il messaggio viene inviato solo quando si verifica la condizione ad esso riferita.

La seconda forma di diagramma di interazione è il *diagramma di collaborazione.* In tale diagramma gli oggetti sono disegnati come icone, fra le quali delle frecce rappresentano i messaggi scambiati in un determinato caso d'uso. Questa volta la sequenza è indicata da numeri posti sui messaggi stessi.

I due tipi di diagrammi sono praticamente equivalenti ed il loro utilizzo è legato al particolare sistema che devono modellare e all'abitudine del progettista nell'usare l'una o l'altra notazione.

### 3.1.6 State diagrams (diagrammi di stato)

I *diagrammi di stato* sono una tecnica consueta per descrivere il comportamento di un sistema. Essi descrivono tutti i possibili stati raggiungibili da un particolare oggetto e come cambia lo stato dell'oggetto in conseguenza degli eventi.

Nella maggior parte delle metodologie object-oriented, si disegnano diagrammi di stato per una singola classe. Ci sono molte forme dei diagrammi di stato, ognuna delle quali ha una semantica leggermente diversa dalle altre: lo stile adottato da UML è basato sullo statechart di David Harel.

I diagrammi di stato sono utili per descrivere il comportamento di un oggetto attraverso più casi d'uso: non sono adatti invece, per descrivere il comportamento di più oggetti coinvolti in una collaborazione.

### 3.1.7 Activity diagrams (diagrammi delle attività)

I *diagrammi delle attività* rappresentano una delle parti più inaspettate dell'UML. A differenza della maggioranza delle altre tecniche che ne fanno parte, questi diagrammi non traggono evidentemente origine da qualche lavoro precedente degli autori dell'UML. Essi combinano idee tratte da tecniche diverse, fra le quali: i diagrammi degli eventi di Jim Odell, le tecniche di modellazione di stato SDL, la modellazione workflow e le reti di Petri.

Questi diagrammi sono particolarmente utili in presenza di workflow e per descrivere processi con una forte componente di modellazione parallela.

### 3.1.8 Physical diagrams (diagrammi fisici)

UML prevede due tipologie di diagrammi che documentano il livello fisico di un'applicazione: i diagrammi di *deployment* e i diagrammi dei *componenti*.

Un *diagramma di deployment* mostra le relazioni fisiche tra i componenti software e hardware del sistema finito. Il diagramma è il posto giusto dove mostrare come vengono instradati gli oggetti e i componenti di un sistema distribuito.

Un *diagramma dei componenti* illustra i vari componenti di un sistema e le loro dipendenze. Un componente rappresenta un modulo di codice. Spesso un componente coincide con un package, ma potrebbe essere differente, dal momento che i componenti rappresentano un'unità fisica di codice. Di conseguenza, una singola classe potrebbe essere presente in più componenti, ma quella classe può essere definita in un solo package.

Le dipendenze fra i componenti mostrano come i cambiamenti apportati ad uno di essi, si ripercuotono sugli altri. C'è una gran varietà di possibili dipendenze, tra cui quelle di comunicazione e quelle di compilazione. Spesso queste dipendenze vengono usate per mostrare in che modo comunicano i componenti.

## 3.2 Il progetto

Presentiamo il progetto realizzato in UML dell'applicazione server-side per la pubblicazione automatica di notizie sul Web, chiamata brevemente *motore di news*.

La modellazione object-oriented permette differenti livelli di approfondimento, addentrandosi sempre di più nel problema. Durante il progetto sono stati messi in luce i differenti aspetti legati alle funzioni principali dell'applicazione, fermandosi laddove la modellazione avrebbe creato strutture eccessivamente complesse, non più adatte ad una comprensione intuitiva del funzionamento.

### 3.2.1 Le specifiche

L'intento è quello di progettare un'applicazione in grado di pubblicare in rete delle notizie composte da un titolo, un corpo del testo ed eventualmente un'immagine e dei link.

Con il termine *pubblicare in rete* s'intende rendere visibili tali notizie in pagine dinamiche create in base a dati estratti da un database.

Lo scopo di tale applicazione è quello permettere a personale a digiuno di HTML di mantenere aggiornata la lista delle notizie semplicemente compilando un form. Le notizie e tutti i dati relativi ad esse, vengono memorizzate in un database ad eccetto delle immagini che vengono memorizzate nel file system.

Per regolare l'accesso a tale sistema è necessaria la gestione delle autorizzazioni degli utenti tramite un'autenticazione attraverso username e password. L'autenticazione si rende necessaria per la presenza di ruoli differenti degli utenti all'interno del sistema.

Sono richieste le funzioni di lettura e composizione di notizie, pubblicazione e cancellazione, recupero delle notizie erroneamente cancellate, gestione degli utenti e dei permessi ad essi accordati.

Inoltre è doveroso fare in modo che coloro che accedono al sistema interagiscano con menù personalizzati che offrano solamente le voci a cui l'utente è autorizzato. Gli utenti possono fare parte di quattro tipologie differenti con ruoli diversi: lettore, creatore, pubblicatore, amministratore di sistema. Nel seguito del progetto saranno descritte le differenze fra le quattro tipologie.

Sotto l'aspetto funzionale, l'applicazione deve garantire l'accesso concorrente di diversi utenti e, naturalmente, garantire l'integrità e la consistenza dei dati.

Tutte le funzioni dell'interfaccia utente devono rispondere ai requisiti di semplicità d'uso e di velocità, che portano il sistema ad essere realmente efficiente ed utilizzabile.

Inoltre deve essere possibile adattare il motore di news alla maggior parte dei DBMS in commercio, possibilmente senza la necessità di effettuare operazioni di riscrittura di codice.

### 3.2.2 Use cases

#### *Utenti*

Nel sistema si delineano quattro tipi di utenti (Actors) che accedono in modo sostanzialmente differente all'applicazione di gestione della news.

?? Creatore
?? Lettore
?? Pubblicatore
?? Superuser

In figura è mostrata la generalizzazione esistente fra gli tutti i tipi di utenti che possono accedere al sistema. Ogni tipologia di utente è rappresentata da un diverso *actor* (attore) che avrà un diverso ruolo nell'utilizzo dell'applicazione.

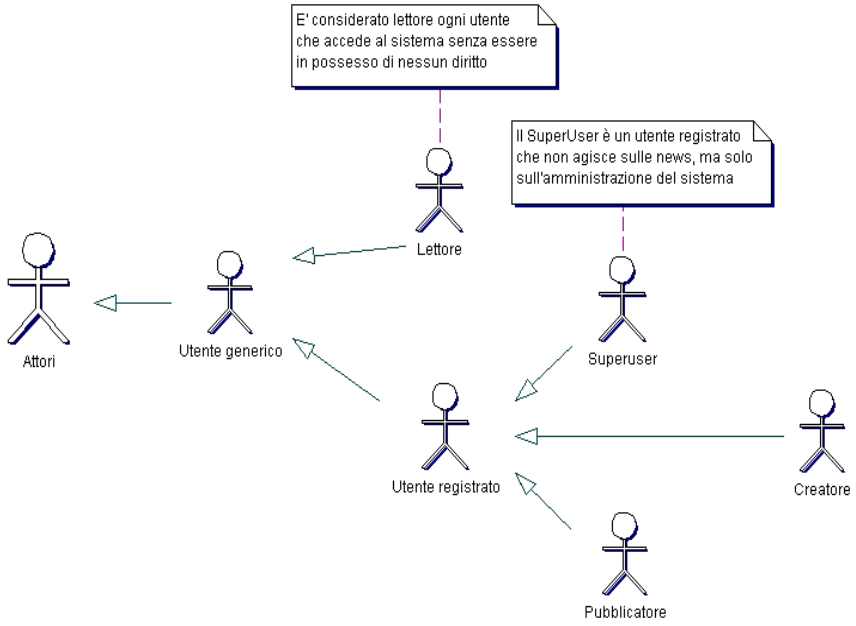


**Figura 11: generalizzazione degli utenti**

Nella realtà, un singolo utilizzatore potrà avere contemporaneamente i diritti di *creatore*, *pubblicatore* e *superuser*, creando una sovrapposizione fra i ruoli.

Questo non deve trarre in inganno in questa fase del progetto, poiché comunque agisca, l'utente agirà di volta in volta facendo parte della tipologia abilitata alla singola funzione.

## *Lettore (Utente generico)*

Appartiene a questa categoria un generico utente che accede al sistema senza effettuare l'operazione di login. Il *lettore* compie una consultazione ed una lettura delle news, cioè è l'utente finale che usufruisce del servizio.

La lettura consiste nella visualizzazione e la consultazione delle news già pubblicate.

1. Il *lettore* accede al servizio di *lettura News* tramite un link dalla Home page.

2. Compare un elenco composto da tutte le news pubblicate, per ognuna delle quali vengono visualizzate le seguenti informazioni:
   - ?? Titolo
   - ?? Data di creazione
   - ?? Creatore
   - ?? Un estratto del testo (ad esempio i primi 25 caratteri)
   - ?? Informazioni riguardanti la notizia, in particolare se contiene link o un'immagine.
3. Il *lettore* seleziona la news a cui è interessato. Solo dopo aver cliccato sulla singola News si può visualizzare il testo completo, i link e l'immagine allegata. L'ordine in cui le News sono presentate nell'elenco è quello decrescente per data di creazione, questo per focalizzare l'attenzione sull'ultima notizia pubblicata.

Tutti gli utenti, anche quelli sprovvisti di username e password appartengono alla categoria dei lettori

## *Creatore*

*Creatore* è colui che redige una news e ne propone la pubblicazione. Per creazione si intende la stesura della news e di tutte le caratteristiche necessarie. La news creata da un utente con i diritti di *creatore* non è immediatamente visibile a tutti. Questo poiché per ottenere visibilità, la notizia deve essere pubblicata. La decisione se pubblicare o meno una news è opera dell'utente *pubblicatore.*

Il *creatore* accede ad un menù con le seguenti funzioni:

1. Lettura delle news già pubblicate.
2. Cancellazione delle proprie news pubblicate e non pubblicate
3. Creazione di una nuova news
4. Logout dal sistema

Selezionando la voce corrispondente alle singola funzione, il creatore si troverà ad interagire con scenari differenti.

## *Pubblicatore*

Il *pubblicatore* si occupa di decidere quali news meritano la possibilità di essere visualizzate dagli utenti lettori e quali, invece, devono essere rimosse dall'elenco poiché non interessanti.

Il pubblicatore accede ad un menù con le seguenti funzioni:

1. Lettura delle news già pubblicate
2. Pubblicazione/cancellazione delle news
3. Logout dal sistema

Il lavoro del *pubblicatore* è quello di visionare le news, pubblicarle, rimandare la decisione o cancellarle definitivamente.

## *Superuser*

Il *Superuser* è da considerarsi come un manutentore del sistema e il suo intervento deve manifestarsi solamente in particolari occasioni in cui è necessario aggiungere o togliere utenti al sistema, recuperare notizie erroneamente cancellate o adattare il sistema ad altri cambiamenti del mondo reale.

Il super utente accede ad un menù con le seguente funzioni:

1. Inserimento nuovo utente
2. Visualizzazione e gestione dei dati degli utenti inseriti
3. Recupero delle notizie erroneamente cancellate da altri utenti
4. Logout dal sistema

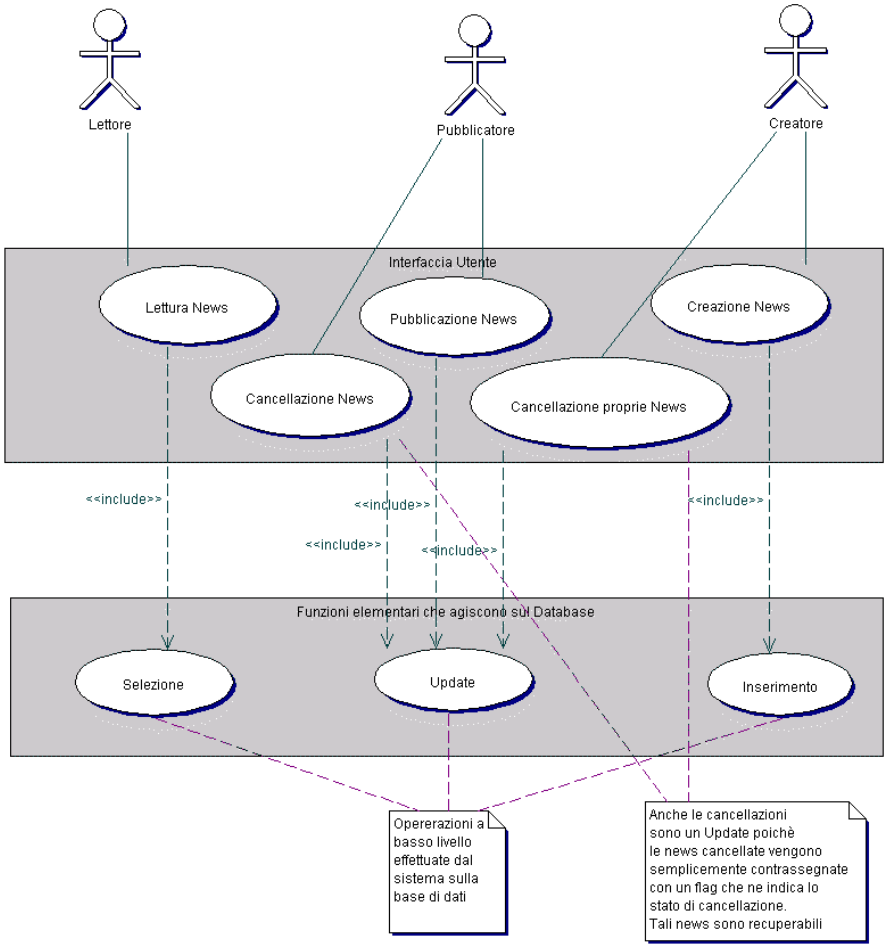


**Figura 12: attori e funzioni sulle news**

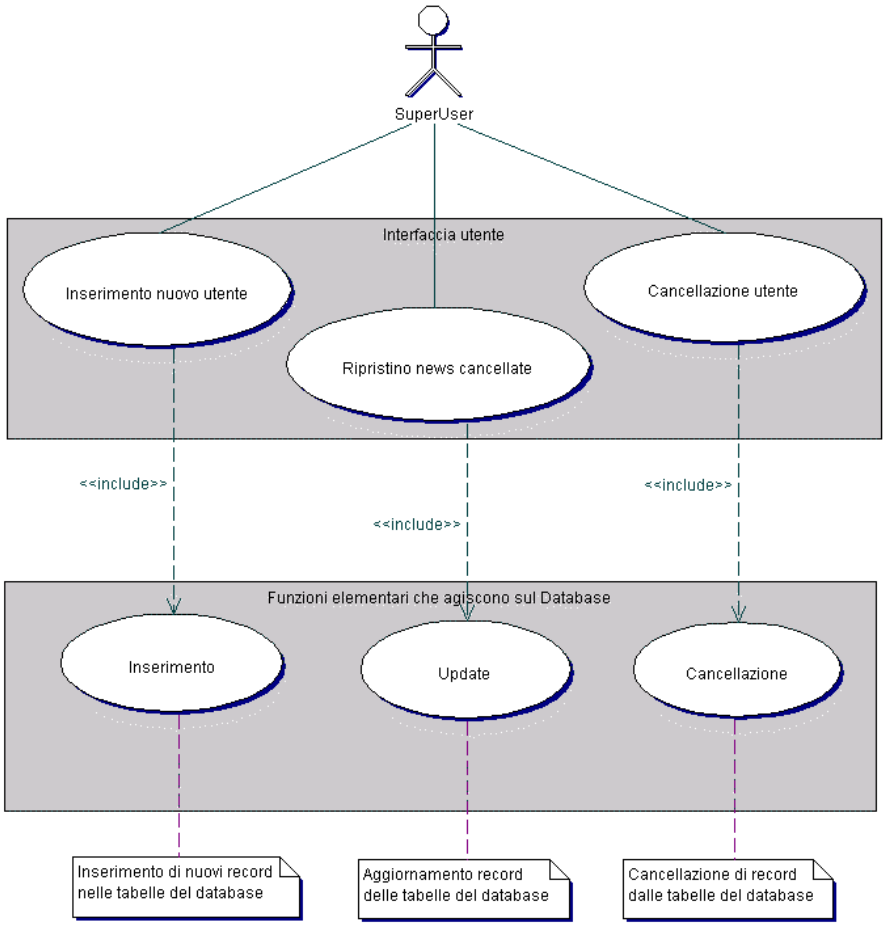


**Figura 13: superuser e funzioni di manutenzione del sistema**

## *Login di utenti registrati*

1. Un utente in possesso di autorizzazioni di qualunque tipo, accede al sistema tramite username e password.
2. L'utente compila due caselle di testo nelle quali vengono richiesti username e password.
3. Se i dati immessi sono corretti l'utente ha accesso ad un menù personalizzato in base ai diritti di cui è in possesso.

**Alternativo:** compilazione incompleta al punto 1.
L'utente viene avvisato della mancata compilazione di uno dei due campi, senza che i dati vengano inviati al server.

**Alternativo:** dati non validi al punto 2.
L'utente viene avvisato dell'inesattezza dei dati e gli viene fornita la possibilità di effettuare un nuovo login.

### *Creazione di una nuova news*

1. Selezione della funzione di Creazione.
2. Il Creatore immette le informazioni richieste nei campi:
   - ?? Tipo della news (solo testo, testo e immagine)
   - ?? Titolo
   - ?? Data di creazione
   - ?? Corpo del testo
   - ?? Eventuali Link correlati alla News
   - ?? Eventuali Immagini allegati alla News
3. Invio dei dati.
4. Il sistema controlla e convalida i dati inseriti.
5. Il sistema informa che la News è stata creata correttamente e la pone nello stato di 'non pubblicata'.

**Alternativo:** dati non validi (punto 4).
Al punto 4 sono state rilevate inesattezze e non i dati non vengono convalidati.
4a. Il sistema segnala le inesattezze presenti.
5a. Ritorno al punto 2.

**Alternativo:** selezione del tipo di news con immagine (punto 2).
Al punto 2 è stato scelto il formato che prevede un'immagine.
3b. Compare una seconda maschera in cui viene gestito l'upload del file nel computer remoto.
4b. Viene data conferma di trasferimento del file.
5b. L'esecuzione riprende dal punto 5.

Dopo la fase di creazione, la news viene inserita nel database nello stato 'non pubblicata', in attesa dell'intervento del *pubblicatore* che deciderà se rimuoverla o pubblicarla.

L'utente *creatore* ha i diritti di cancellazione su tutte e solo le news scritte da lui, anche se queste sono già state pubblicate.

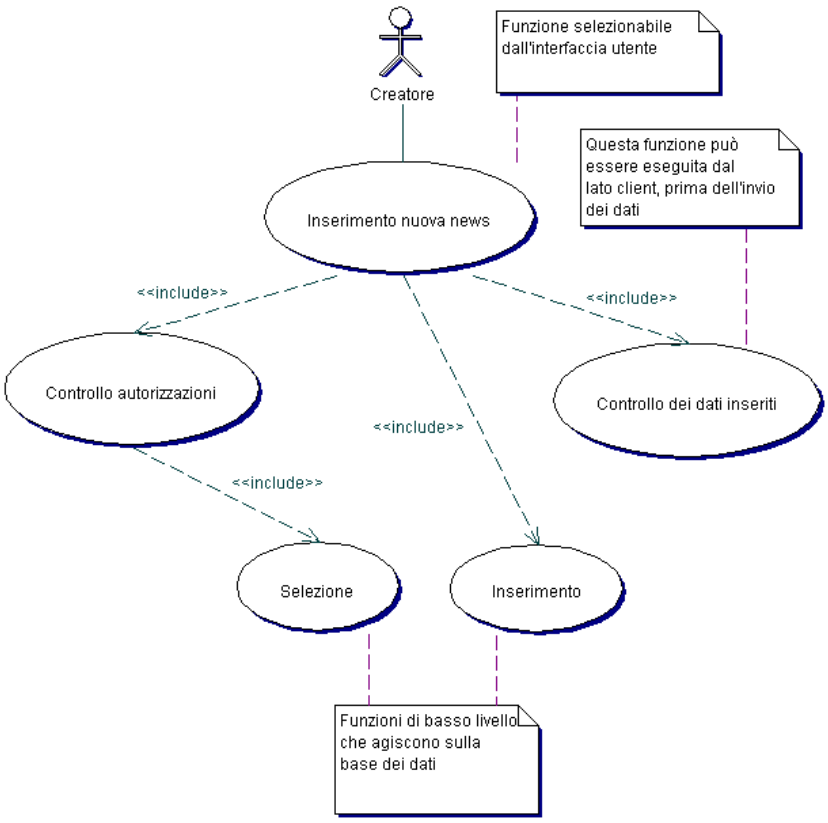


**Figura 14: inserimento di una nuova news**

Quando il creatore accede alla funzione di creazione di una nuova news, viene eseguito un controllo delle autorizzazioni per evitare che a questa funzione possano accedere 'abusivamente' anche utenti non registrati o non in possesso dei diritti necessari.

La funzione che si occupa della correttezza dei dati inseriti, è un JavaScript che gira sul lato client. Questa scelta è stata effettuata per

evitare di impegnare le preziose risorse del server e per fare in modo che ad esso giungano solo i dati correttamente inseriti dall'utente. Possibili errori di inserimento potrebbero sorgere qualora vengano lasciati vuoti alcuni campi relativi a dati indispensabili.

## *Pubblicazione*

1. Selezione della funzione di pubblicazione.
2. Visualizzazione elenco di tutte le news (pubblicate e non) di tutti gli autori. Per ogni news è quindi visualizzato:
   - ?? Titolo
   - ?? Data di creazione
   - ?? Creatore
   - ?? Pubblicatore
   - ?? Un estratto del testo (ad esempio i primi 25 caratteri)
   - ?? Informazioni riguardanti la notizia
3. Selezione tramite checkbox delle news su cui agire
4. Selezione della funzione di pubblicazione
5. Messaggio di conferma della pubblicazione
6. Visualizzazione della lista delle news aggiornata dopo l'operazione di pubblicazione

**Alternativo:** selezione della news su cui agire non facendo uso della check box (punto 3).

3a. La news è selezionata cliccando sul titolo.
4a. La news viene mostrata espansa a tutto schermo.
5a. E' possibile pubblicare la news cliccando sull'apposita icona.
6a Messaggio di conferma della pubblicazione.
7a. Ritorno alla lista delle news.

Il *pubblicatore* agisce sia sulle news non ancora pubblicate che su quelle già pubblicate attraverso una pagina simile alla pagina vista dal *lettore*, ma in cui compaiono anche le news create (quindi memorizzate nel database) ma in attesa di pubblicazione.

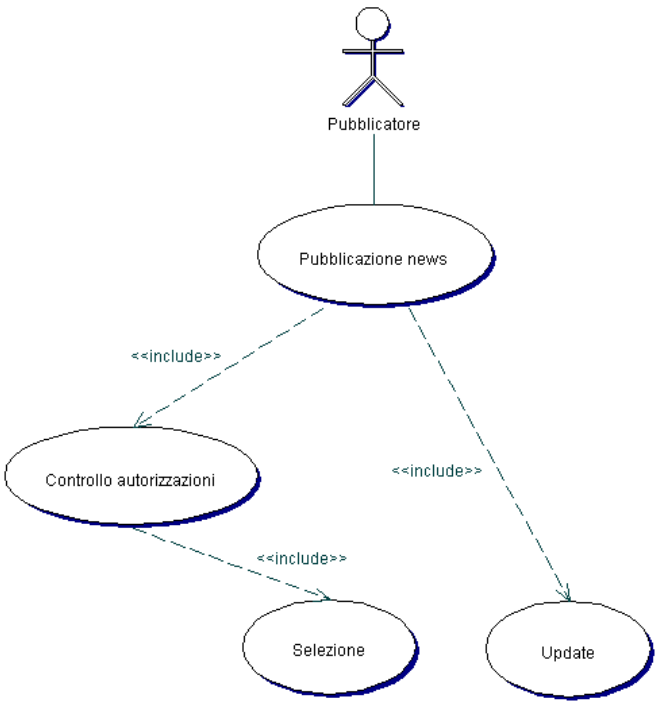


**Figura 15: pubblicazione di una News**

## *Cancellazione di una news da parte del pubblicatore*

L'utente con i diritti di pubblicazione ha la possibilità di cancellare le notizie.

1. Selezione della funzione di pubblicazione dal menù.
2. Visualizzazione elenco di tutte le news (pubblicate e non) di tutti gli autori.
3. Selezione della news da cancellare cliccando sul titolo.
4. La news viene mostrata a tutto schermo.
5. Si cancella la news cliccando sull'apposita icona.
6. Messaggio di conferma della cancellazione.
7. Ritorno al menù principale.

## *Cancellazione di una news da parte del creatore*

L'utente in possesso dei diritti di creazione ha la possibilità di cancellare solamente le proprie notizie.

1. Selezione della funzione di cancellazione di una propria news.
2. Visualizzazione della lista delle proprie news, pubblicate e non pubblicate.
3. Selezione della news da cancellare cliccando sul titolo.
4. La news viene mostrata a tutto schermo.
5. Si cancella la news cliccando sull'apposita icona.
6. Messaggio di conferma della cancellazione.
7. Ritorno al menù principale.

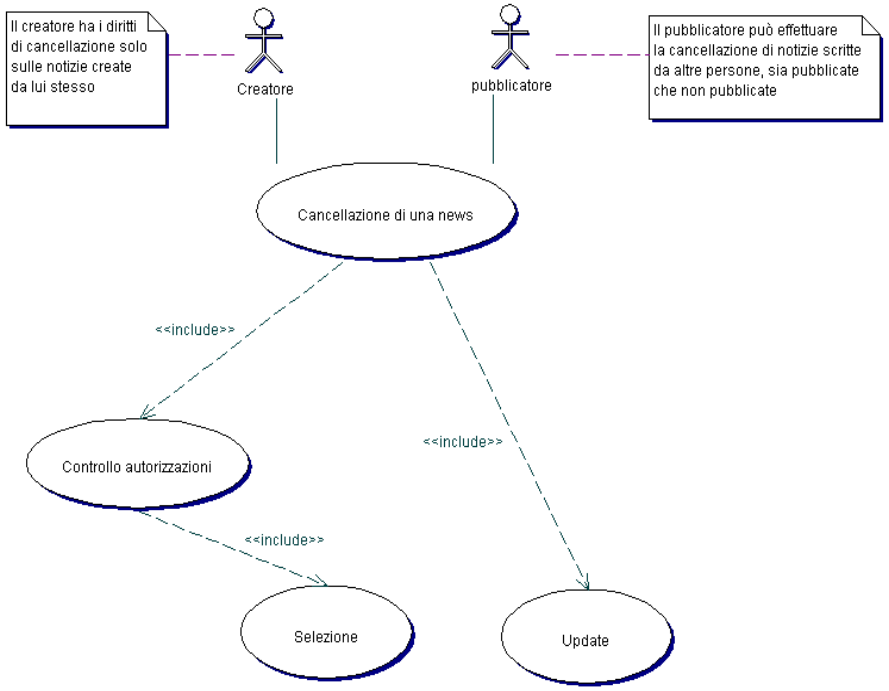


**Figura 16: cancellazione di una news**

*Inserimento nuovo utente*

1. Dal menù principale si accede alle funzioni di amministrazione del sistema.
2. Compare un nuovo menù che mostra le funzioni a cui è abilitato il superuser.
3. Si seleziona la funzione di Inserimento di un nuovo utente.
4. Compare una maschera dove è possibile inserire i del nuovo utente.
5. Invio dei dati.
6. Controllo dell'esattezza dei dati inviati.
7. Messaggio di conferma dell'inserimento del nuovo utente.

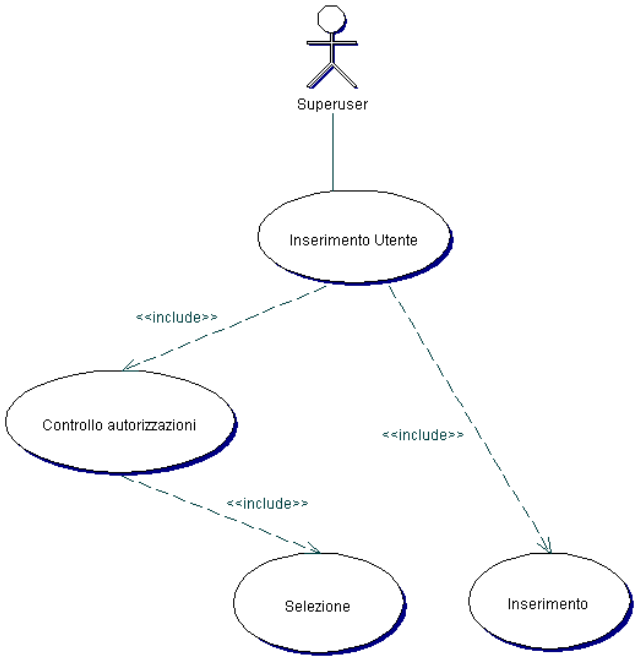


**Figura 17: inserimento di un nuovo utente**

### *Recupero di news cancellate*

Solamente l'utente in possesso dei diritti di *superuser* può accedere a tale funzione.

1. Dal menù principale si accede alle funzioni di amministrazione del sistema.
2. Compare un nuovo menù che mostra le funzioni a cui è abilitato il superuser.
3. Si seleziona la funzione di recupero delle news cancellate.
4. Compare la lista delle news che sono state precedentemente cancellate.
5. Si seleziona la news da recuperare cliccando sul titolo.
6. La news viene espansa a tutto schermo.
7. Si recupera la news cliccando sull'apposita icona.
8. Messaggio di conferma dell'avvenuto recupero.

## 3.2.3 Class diagrams

### *Classe News*

?? Creatore: username del creatore della news
?? Titolo: titolo della news
?? Corpo: corpo del testo della news
?? IdNews: identificativo univoco della news all'interno del sistema
?? Pubblicata: indicazione dello stato di pubblicata/non pubblicata della news
?? Pubblicatore: username del pubblicatore della news
?? Cancellata: indicazione dello stato di cancellata/non cancellata
?? Firma: stringa contenente la firma del creatore
?? Data: data di creazione della notizia
?? Tipo: indicazione della presenza o meno di un'immagine allegata alla notizia

### *Classe Link*

?? Id: identificativo della notizia a cui appartiene il link

?? Link: URL della pagina lincata
?? Descrizione: breve descrizione che viene visualizzata

### *Classe Immagine*

?? Id: identificativo della news a cui appartiene l'immagine
?? Nome: nome relativo del file dell'immagine allegata

### *Classe Utente*

?? Username: username univoco dell'utente all'interno del sistema
?? Password: password personale di accesso al sistema

### *Classe Funzione*

?? Username: username dell'utente correlato alle funzioni dell'istanza
?? Scrivi: diritti di creazione delle news
?? Pubblica: diritti di pubblicazione delle news
?? Autorizza: diritti di superuser

### *Relazione fra Utenti e Funzioni*

Il diagramma va inteso a livello di implementazione e mostra il meccanismo con il quale siano stati differenziati i vari tipi di utenti registrati.

L'attributo firma della classe Utente rappresenta la firma che viene visualizzata sulla notizia e rimane fissa per ogni utente con i diritti di creazione.

Nel diagramma viene utilizzata una modellazione che può risultare in contrasto con quanto descritto negli use cases precedenti, nei quali era rappresentata una generalizzazione degli utenti come indicato nel diagramma use case. In questo caso è però preferibile, a livello di implementazione, raggruppare tutti gli utenti registrati in una stessa classe e metterli in relazione con delle funzioni. Nella realizzazione pratica la molteplicità 1:1 è garantita dal fatto che lo username risulta univoco sia per le istanze di Utente, sia per le istanze di Funzione.

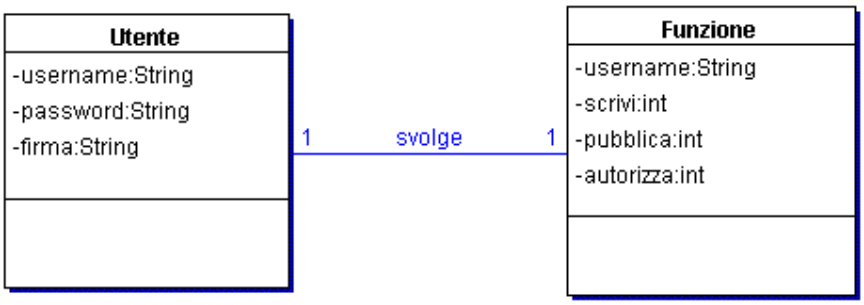


**Figura 18: relazione tra classe Utente e classe Funzione**

Come già evidenziato nei diagrammi di casi d'uso, non è escluso che uno stesso attore possa interpretare il ruolo di tutti o alcuni actors.

Questa possibilità è giustificata dall'esigenza pratica di concentrare la carica di creatore di news, pubblicatore e manutentore del sistema, nella mani di un solo soggetto, soprattutto per piccole realtà in cui viene applicato il motore di news.

Mantenendo traccia delle autorizzazioni nel modo indicato in figura, risulta naturale poter sovrapporre i ruoli.

Infatti se gli attributi *scrivi*, *pubblica* e *autorizza* assumono valore 1, allora la corrispondente azione è autorizzata, facendo assumere all'attore il corrispondente ruolo. Diversamente, se assumono il valore 0 la corrispondente azione è vietata.

### *La presenza di Immagine e Link in una News*

Come anticipato, una news può essere corredata di un'immagine e di un numero imprecisato di link.

Il diagramma, da leggersi dal punto di vista implementativo, mostra come siano state definite delle classi Link e Immagini e come siano in relazione con la classe News.

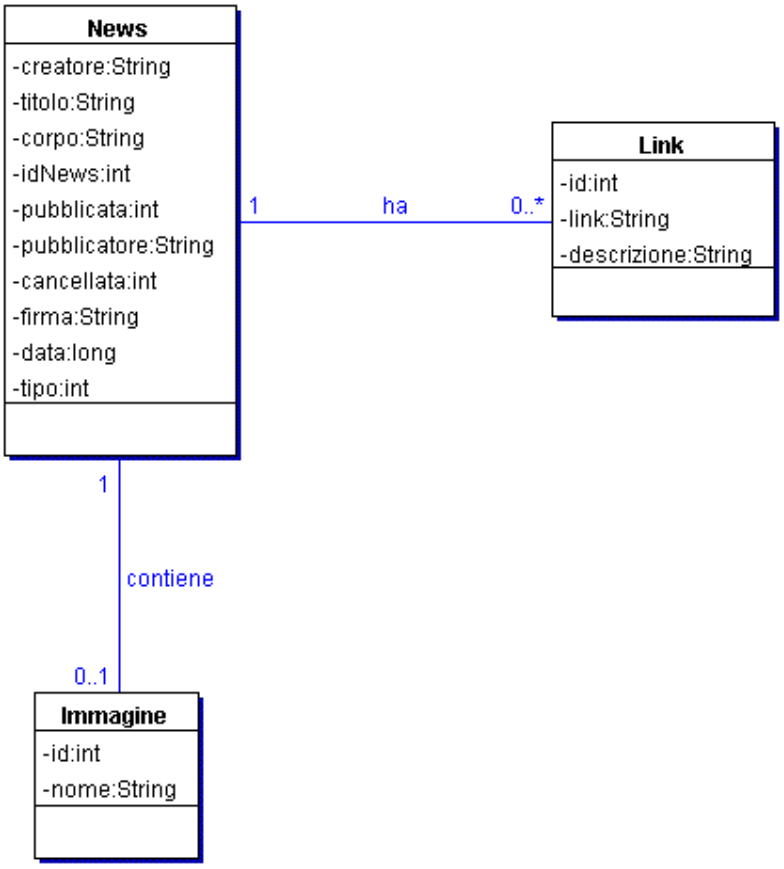


**Figura 19: relazioni fra News, Immagini e Link**

### 3.2.4 Sequence diagrams

*Login di un utente registrato*

L'utente generico (il lettore) accede al sistema liberamente, senza che venga richiesta nessuna autenticazione, ma le sue funzioni sono ridotte alla sola lettura delle notizie già pubblicate.

Per l'utente che sia in possesso di qualche diritto, è indispensabile effettuare una fase di login.

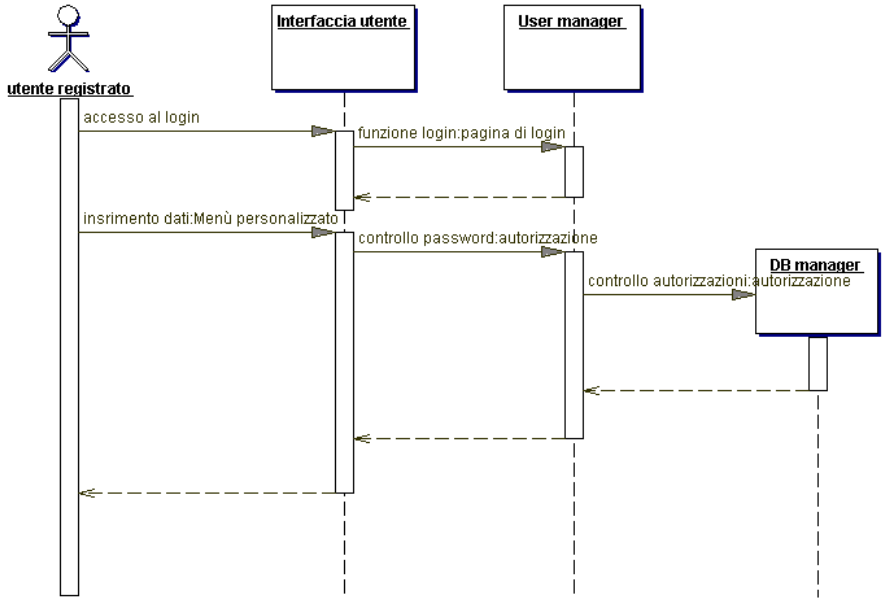


**Figura 20: fase di login di un utente registrato**

Nel caso di login effettuato correttamente, la pagina di risposta consiste nella pagina con il menù personalizzato in base alle funzioni a cui è abilitato l'utente.

Nel caso di login non corretto, la pagina risposta è un messaggio che informa dell'errore commesso.

## *Scrittura di una nuova notizia*

Solo l'utente con i diritti di creazione può accedere alla funzione di composizione di una nuova notizia. Tuttavia, per evitare tentativi di forzatura della sicurezza del sistema, ogni volta che un utente prova ad accedere a tale funzione, viene effettuato un controllo sui diritti.

Questa operazione rappresenta un ulteriore controllo, poiché la funzione di composizione è selezionabile da interfaccia utente solamente dagli utenti in possesso dei diritti. Nonostante ciò non è difficile forzare il sistema in modo da accedere ugualmente a tale funzione. Per questo motivo viene effettuato l'ulteriore controllo.

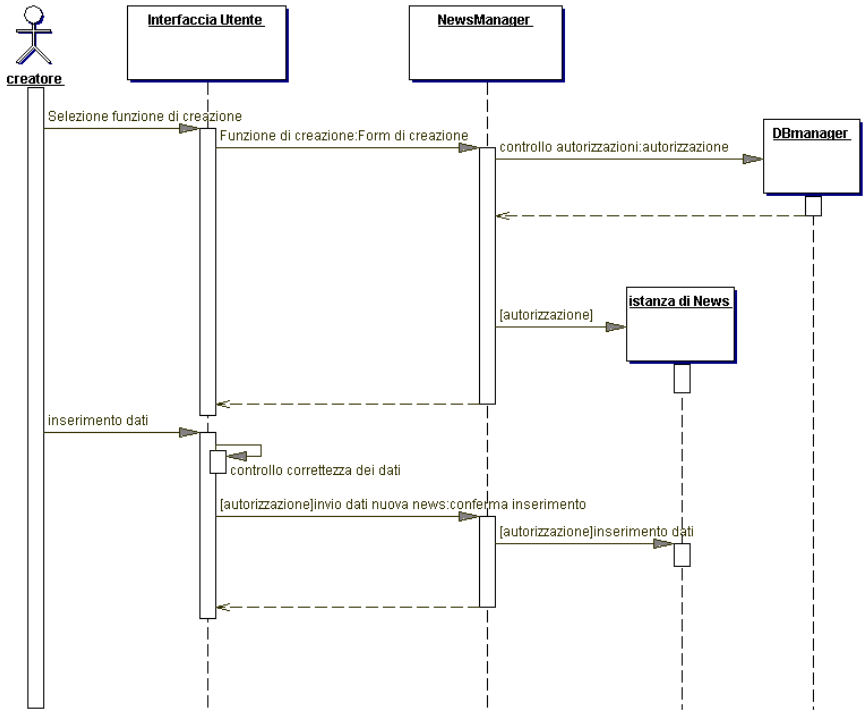


**Figura 21: sequenza di creazione di una nuova notizia**

L'oggetto News Manager si occupa di effettuare le operazioni relative alle news. L'oggetto DB manager gestisce gli accessi alla base di dati e in questo caso l'interroga per verificare l'autorizzazione dell'utente.

## *Pubblicazione o cancellazione di una notizia*

Anche in questo caso, viene controllata la presenza delle autorizzazioni necessarie. Il pubblicatore può scegliere se pubblicare o cancellare la news.

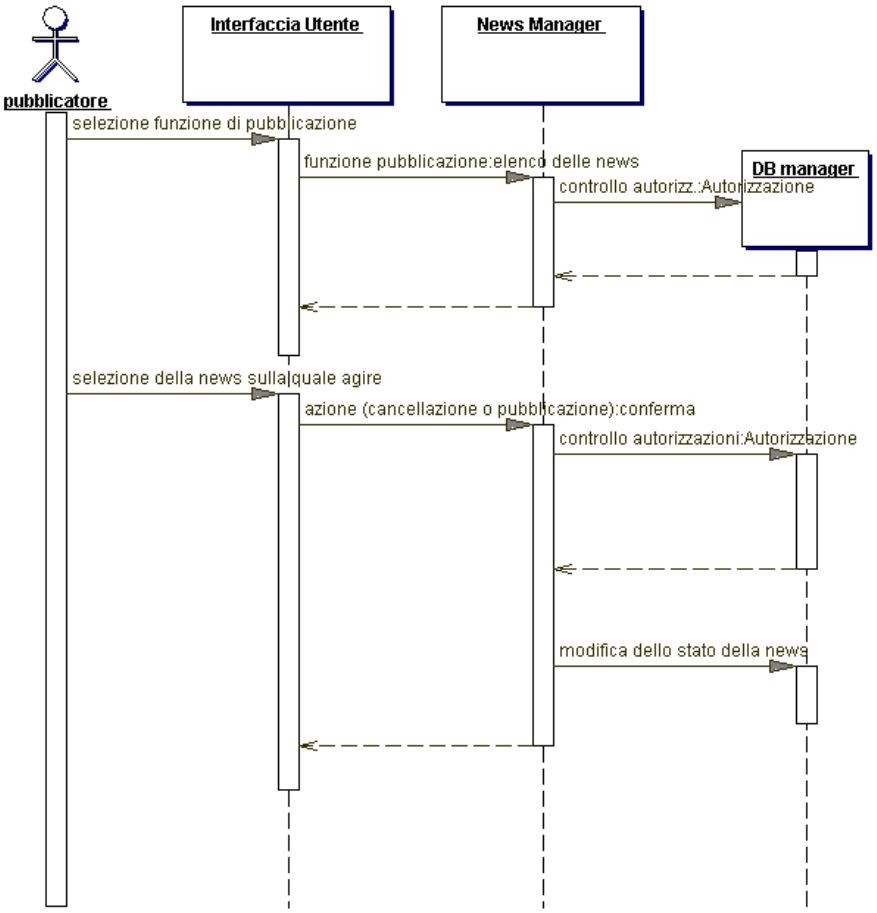


**Figura 22: operazioni di pubblicazione o cancellazione sulla news**

Queste due operazioni sono precedute entrambe dalla selezione della news su cui agire che comporta un'espansione della notizia a tutta pagina, per una corretta visualizzazione. Questo, per semplicità, non è stato rappresentato nel diagramma.

## *Lettura di una news*

La lettura è permessa ad ogni utente che accede al sito, senza che possieda nessun diritto. Nel diagramma viene mostrata la sequenza delle fasi coinvolte nell'operazione di lettura.

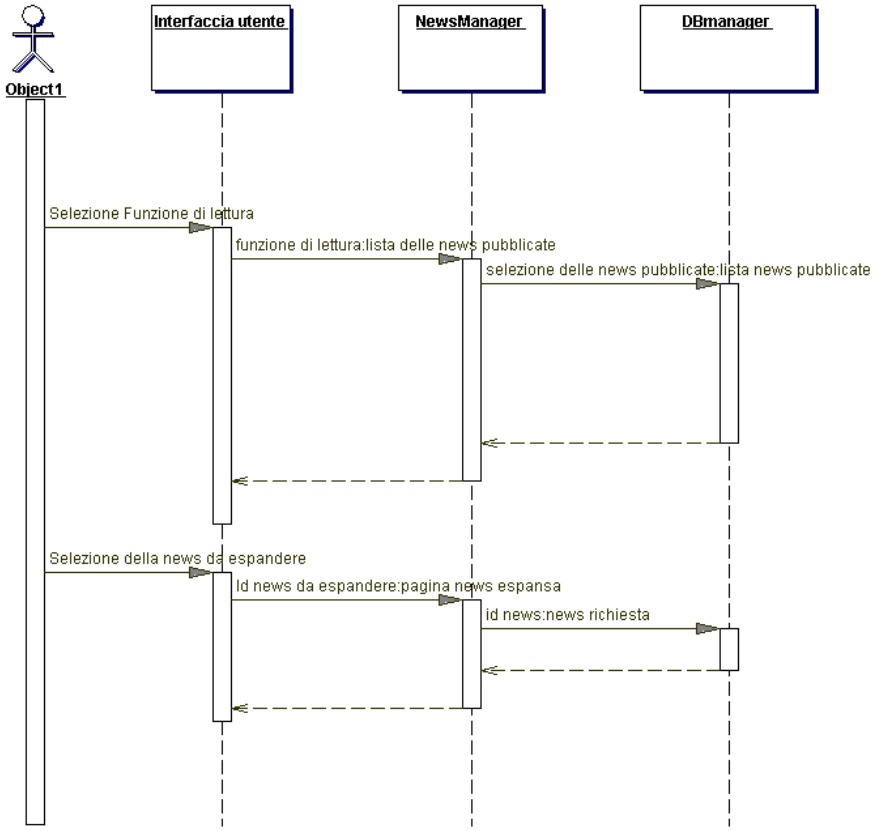


**Figura 23: lettura di una notizia**

*Inserimento di un nuovo utente*

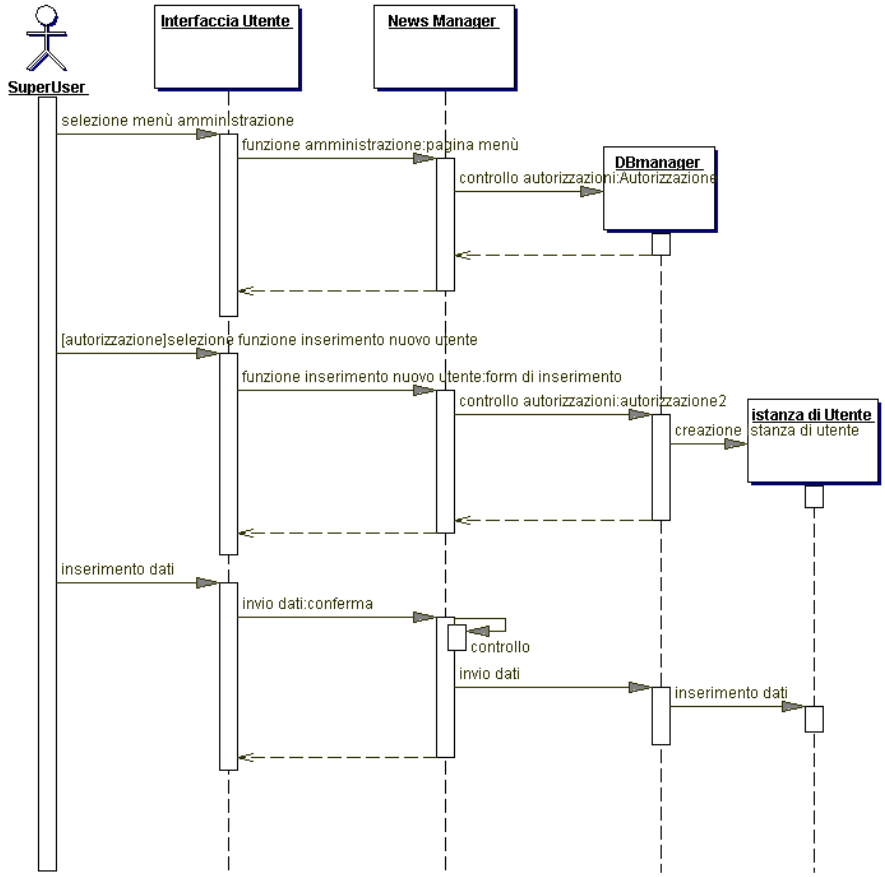


**Figura 24: inserimento di un nuovo utente**

*Recupero news cancellate*

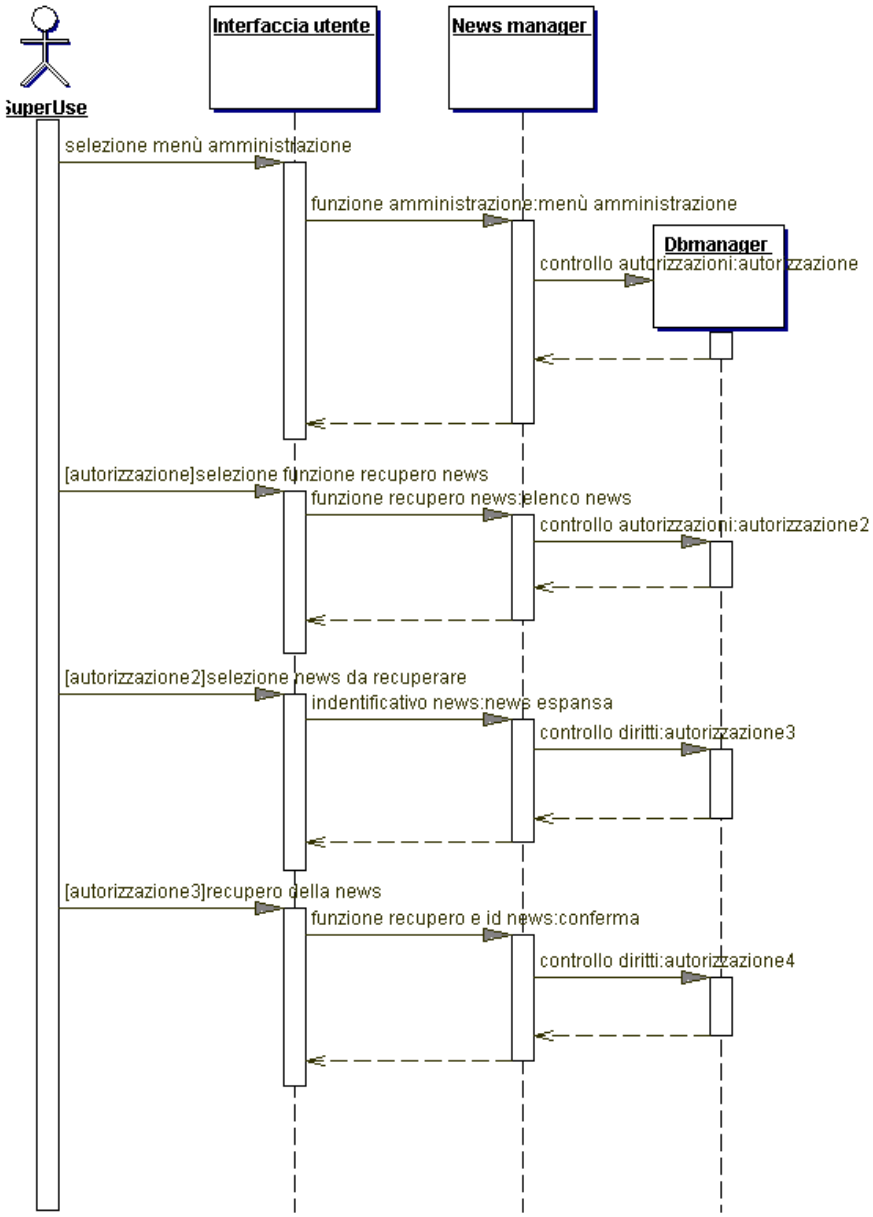


**Figura 25: recupero di una notizia precedentemente cancellata**

## *Analogie fra gli i diagrammi di sequenza*

Nei precedenti paragrafi abbiamo presentato alcuni diagrammi di sequenza relativi alle funzioni principali del motore di news. In tutti i diagrammi compaiono tre oggetti che si scambiano messaggi:

?? Interfaccia Utente
?? News manager o User manager
?? DB manager

L'interfaccia utente rappresenta la parte di programma che si occupa della visualizzazione. L'utente interagisce solo con essa e da essa trae le informazioni richieste.

Il news manager è quella parte dell'applicazione che si occupa di eseguire le operazioni riguardanti le notizie. È il motore dell'applicazione in cui sono contenuti gli algoritmi che eseguono le funzioni selezionabili dall'interfaccia utente.

Lo user manager compie la stessa funzione del news manager, ma è relativo alle operazioni riguardanti gli utenti.

Il DB manager compie operazioni direttamente sulla base di dati, come query di selezione, inserimento e cancellazione record e aggiornamento dei dati.

Già da i primi diagrammi di caso d'uso, si potevano individuare alcune funzioni denominate 'di basso livello' che rappresentano quelle funzioni svolta dal DB manager.

Ripensando alla struttura raccomandata dal pattern strutturale MVC, constatiamo una perfetta analogia fra Model-View-Controller e News Manager-Interfaccia Utente-DB manager.

Questa analogia deriva da una corretta applicazione della separazione funzionale, fin dalle prime fasi del progetto e da un affinamento di questa filosofia attraverso i passi della modellazione.

### 3.2.5 State diagrams

*News*

Il diagramma mostra tutti gli stati attraverso i quali può passare un oggetto della classe News.

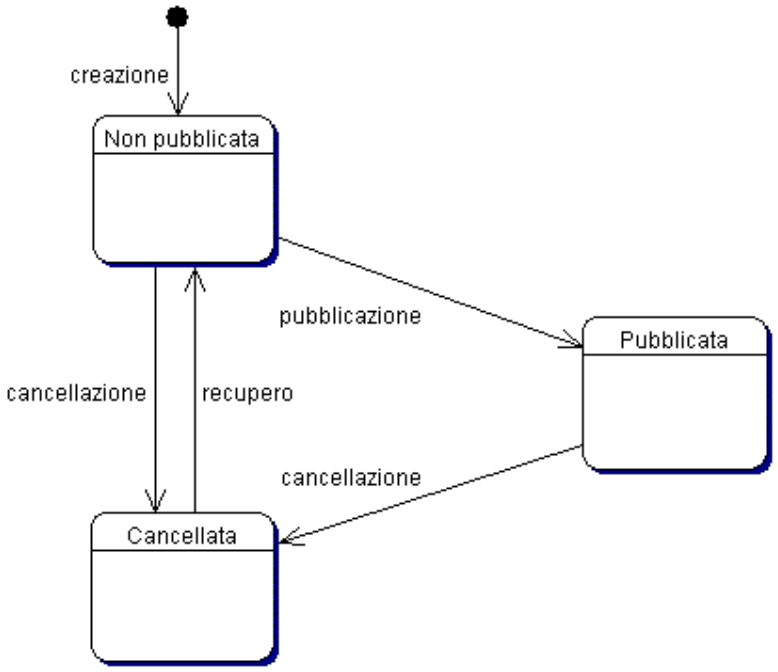


**Figura 26: possibili stati di un'istanza della classe News**

*Utente (accesso al sistema)*

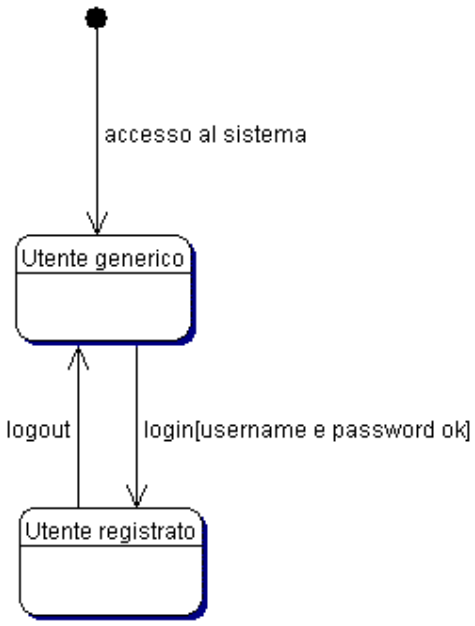


**Figura 27: Possibili stati di un'istanza della classe Utente**

### 3.2.6 Activity diagrams

*Creazione news*

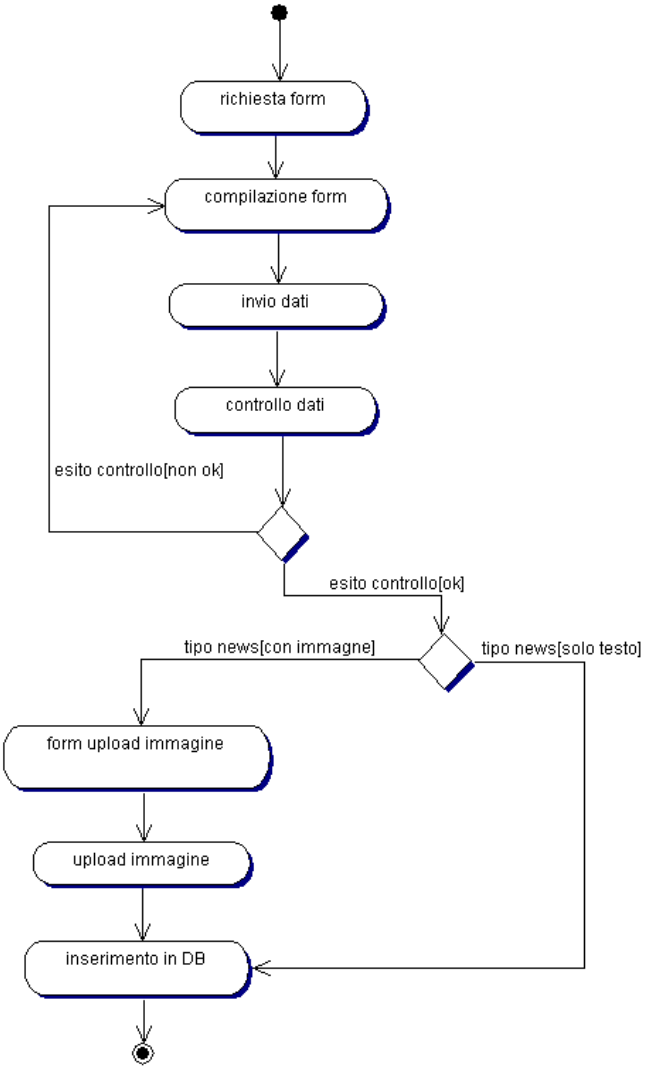


**Figura 28: Attività di creazione di una news**

*Pubblicazione o cancellazione di una news*

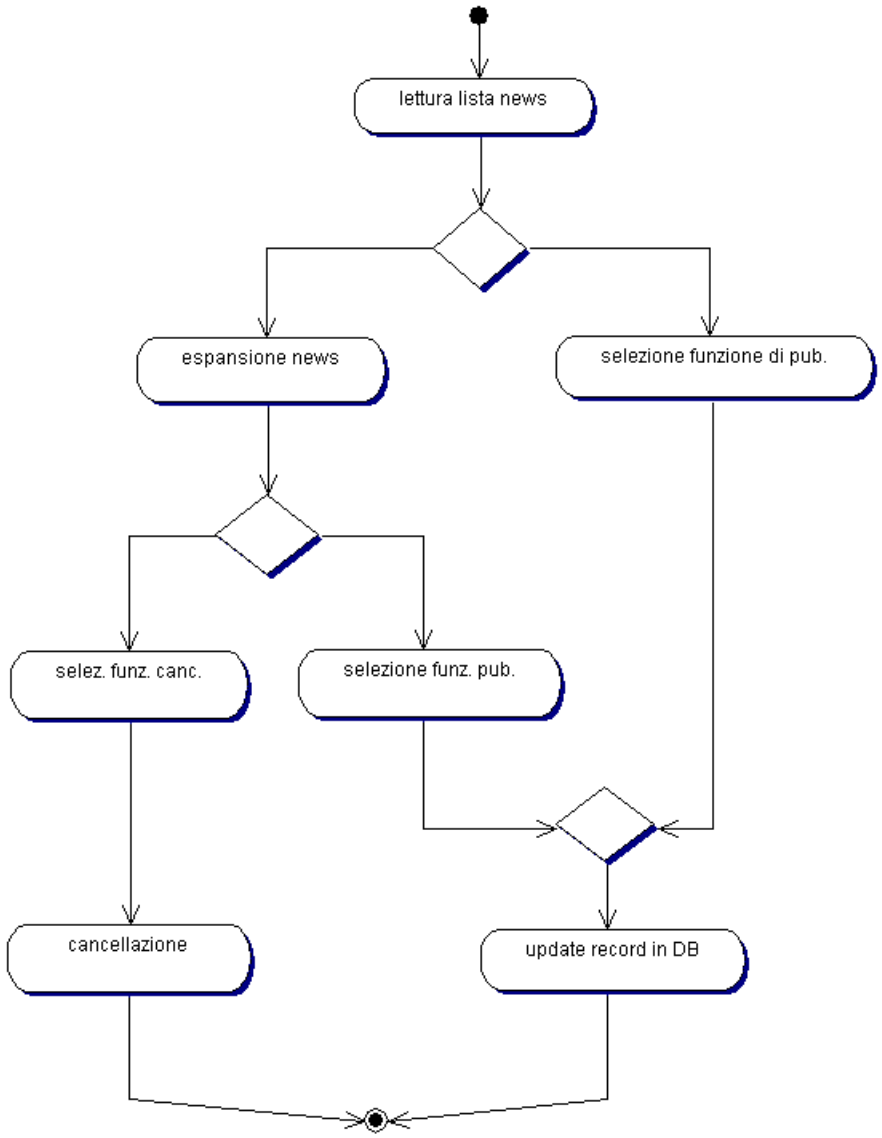


**Figura 29: pubblicazione o cancellazione di una news**

# Capitolo 4

# L’implementazione

In questo capitolo saranno trattate le problematiche relative alla realizzazione pratica del prototipo del motore di news e verrà descritta la soluzione adottata e le possibili soluzioni alternative, partendo dalla parte più lontana dall’utente ed avvicinandosi ad esso fino ad arrivare alle pagine JSP.

## 4.1 I dati

### 4.1.2 Corrispondenza fra classi e tabelle

La struttura della base dei dati è molto semplice e lineare. Il numero delle tabelle create è limitato e la loro struttura ricalca esattamente quella delle classi presentate nel class diagram.

In questo modo è stata creata una precisa corrispondenza fra i record delle tabelle e le istanze delle classi maneggiate durante il programma.

Riportiamo le tabelle in cui sono memorizzati i dati nel DBMS e il listato delle classi Java. I nomi dei campi sottolineati rappresentano la *chiave* per la tabella in esame.

*News*

- Creatore: username del creatore della news
- Titolo: titolo della news
- Corpo: corpo del testo della news
- IdNews: identificativo univoco della news all’interno del sistema
- Pubblicata: indicazione dello stato di pubblicata/non pubblicata della news

- Pubblicatore: username del pubblicatore della news
- Cancellata: indicazione dello stato di cancellata/non cancellata
- Firma: stringa contenente la firma del creatore
- Data: data di creazione della notizia
- Tipo: indicazione della presenza o meno di un'immagine allegata alla notizia

| Nome del campo | Tipo di dato |
|---|---|
| IdNews | intero lungo |
| creatore | testo |
| titolo | testo |
| corpo | testo |
| pubblicata | intero {0,1} |
| pubblicatore | testo |
| cancellata | intero {0,1} |
| firma | testo |
| data | intero lungo |
| tipo | intero {0,1} |

News.java

```
package news;

import java.sql.*;

public class News implements RecordPrototype {

	private String creatore;
	private String titolo;
	private String corpo;
	private int idNews;
	private int pubblicata;
	private String pubblicatore;
	private int cancellata;
	private String firma;
	private long data;
	private int tipo;

public News() {
	super();
}

public java.lang.String getCorpo() {
	return corpo;
}

public String getCreatore() {
	return creatore;
```

```
}

public long getData() {
	return data;
}

public java.lang.String getFirma() {
	return firma;
}

public int getIdNews() {
	return idNews;
}

public RecordPrototype getInstance() {

	News n = new News();
	return n;

}

public java.lang.String getPubblicatore() {
	return pubblicatore;
}

public int getTipo() {
	return tipo;
}

public java.lang.String getTitolo() {
	return titolo;
}

public int isCancellata() {
	return cancellata;
}

public int isPubblicata() {
	return pubblicata;
}

public void setCancellata(int newCancellata) {
	cancellata = newCancellata;
}

public void setCorpo(java.lang.String newCorpo) {
	corpo = newCorpo;
}

public void setCreatore(String newCreatore) {
	creatore = newCreatore;
}

public void setData(int newData) {
	data = newData;
}
public void setData(long newData) {
	data = newData;
}
```

```
public void setFirma(java.lang.String newFirma) {
	firma = newFirma;
}

public void setIdNews(int newIdNews) {
	idNews = newIdNews;
}

public void setPubblicata(int newPubblicata) {
	pubblicata = newPubblicata;
}

public void setPubblicatore(java.lang.String newPubblicatore) {
	pubblicatore = newPubblicatore;
}

public void setTipo(int newTipo) {
	tipo = newTipo;
}

public void setTitolo(java.lang.String newTitolo) {
	titolo = newTitolo;
}

public void setValues(ResultSet rs) {
	try{

		this.setCreatore(rs.getString("creatore"));
		this.setTitolo(rs.getString("titolo"));
		this.setCorpo(rs.getString("corpo"));
		this.setIdNews(rs.getInt("IDnews"));
		this.setPubblicata(rs.getInt("pubblicata"));
		this.setPubblicatore(rs.getString("pubblicatore"));
		this.setCancellata(rs.getInt("cancellata"));
		this.setFirma(rs.getString("firma"));
		this.setData(rs.getLong("data"));
		this.setTipo(rs.getInt("tipo"));

	}


	catch (SQLException sqle) {

		System.err.println(sqle.getMessage());
	}


}
}
```

**Figura 30: listato del codice della calsse News**

## *Utente*

- ?? Username: username dell’utente, univoco all’interno del sistema
- ?? Password: password di autenticazione
- ?? Firma: firma che viene visualizzata nelle notizie

| Nome del campo | Tipo di dato |
|---|---|
| username | testo |
| password | testo |
| firma | testo |

Utente.java

```
package news;

import java.sql.*;

public class Utente implements RecordPrototype{

	private String username;
	private String password;
	private String firma;

public Utente() {
	super();
}

public java.lang.String getFirma() {
	return firma;
}

public RecordPrototype getInstance() {

	Utente utente = new Utente();
	return utente;

	}

public java.lang.String getPassword() {
	return password;
}

public java.lang.String getUsername() {
	return username;
}

public void setFirma(java.lang.String newFirma) {
	firma = newFirma;
}

public void setPassword(java.lang.String newPassword) {
	password = newPassword;
}

public void setUsername(java.lang.String newUsername) {
	username = newUsername;
}

public void setValues(ResultSet rs) {

	try{
```

```
                this.setUsername(rs.getString("username"));
                this.setPassword(rs.getString("password"));
                this.setFirma(rs.getString("firma"));

        }

        catch (SQLException sqle) {

                System.err.println(sqle.getMessage());
                System.err.println("Eccezione SQLException: Metodo
setValues, Classe: Utente");
        }

}
}
```

**Figura 31: listato del codice della classe Utente**

## *Immagine*

- Id: identificativo della news alla quale è allegata l'immagine
- Nome: nome relativo dell'immagine all'interno del file system

| Nome del campo | Tipo di dato |
|---|---|
| id | Intero lungo |
| nome | testo |

Immagine.java

```
package news;

import java.sql.*;

public class Immagine implements RecordPrototype{

        private int id;
        private String nome;

public Immagine() {
        super();
}

public int getId() {
        return id;
}

public RecordPrototype getInstance() {

        Immagine img = new Immagine();
        return img;
```

```
}

public java.lang.String getNome() {
	return nome;
}

public void setId(int newId) {
	id = newId;
}

public void setNome(java.lang.String newNome) {
	nome = newNome;
}

public void setValues(ResultSet rs) {

	try{
		this.setId(rs.getInt("id"));
		this.setNome(rs.getString("nome"));
	}
	catch (SQLException sqle){
		System.err.println(sqle.getMessage());
	}


}
}
```

**Figura 32: listato del codice della classe Immagne**

## *Link*

- ?? Id: identificativo della notizia a cui appartiene il link
- ?? Link: URL della pagina lincata
- ?? Descrizione: breve descrizione che viene visualizzata

| Nome del campo | Tipo di dato |
|---|---|
| id | intero lungo |
| link | testo |
| descrizione | testo |

Link.java

```
package news;

import java.sql.*;

public class Link implements RecordPrototype{

	private int id;
```

```
        private String link;
        private String descrizione;

public Link() {
        super();
}

public java.lang.String getDescrizione() {
        return descrizione;
}

public int getId() {
        return id;
}

public RecordPrototype getInstance() {

        Link l = new Link();
        return l;

}

public java.lang.String getLink() {
        return link;
}

public void setDescrizione(java.lang.String newDescrizione) {
        descrizione = newDescrizione;
}

public void setId(int newId) {
        id = newId;
}

public void setLink(java.lang.String newLink) {
        link = newLink;
}

public void setValues(ResultSet rs) {

try{

                this.setId(rs.getInt("id"));
                this.setLink(rs.getString("link"));
                this.setDescrizione(rs.getString("descrizione"));

        }
        catch (SQLException sqle) {

                System.err.println(sqle.getMessage());
        }

}
}
```

**Figura 33: listato del codice della classe Link**

*Funzione*

- Username: username dell’utente a cui appartengono le funzioni
- Scrivi: permesso di creazione di una nuova news
- Pubblica: permesso di pubblicazione
- Autorizza: permesso di superuser

| Nome del campo | Tipo di dato |
|---|---|
| username | testo |
| scrivi | Intero {0,1} |
| pubblica | Intero {0,1} |
| autorizza | Intero {0,1} |

Funzione.Java

```
package news;

import java.sql.*;

public class Funzione implements RecordPrototype {

	private String username;
	private int scrivi;
	private int pubblica;
	private int autorizza;

public Funzione() {
	super();
}

public RecordPrototype getInstance() {

	Funzione f = new Funzione();
	return f;

}
public java.lang.String getUsername() {
	return username;
}

public int isAutorizza() {
	return autorizza;
}

public int isPubblica() {
	return pubblica;
}

public int isScrivi() {
	return scrivi;
}
```

```
public void setAutorizza(int newAutorizza) {
	autorizza = newAutorizza;
}

public void setPubblica(int newPubblica) {
	pubblica = newPubblica;
}

public void setScrivi(int newScrivi) {
	scrivi = newScrivi;
}

public void setUsername(java.lang.String newUsername) {
	username = newUsername;
}

public void setValues(ResultSet rs) {

	try{
		setUsername(rs.getString("username"));
		setScrivi(rs.getInt("scrivi"));
		setPubblica(rs.getInt("pubblica"));
		setAutorizza(rs.getInt("autorizza"));
	}


	catch (SQLException sqle) {

		System.err.println(sqle.getMessage());
	}

}
}
```

**Figura 34: listato del codice della classe Funzione**

## *Contatori*

| Nome del campo | Tipo di dato |
|---|---|
| id | intero |
| valore | intero lungo |

Questa è la tabella utilizzata per ottenere un intero incrementato di una unità rispetto al precedente valore. L’algoritmo di accesso a tale tabella prevede, ad ogni accesso, l’incremento di una unità del campo valore. La tabella contiene un solo record, al quale si accede tramite il valore di chiave.

### 4.1.3 Lo scambio di dati

Con la suddivisione dell’applicazione in moduli distribuiti in tre livelli differenti, diventa importante descrivere le modalità con cui le diverse parti del software memorizzano e scambiano le informazioni.

La prima preoccupazione è rappresentata dai dati contenuti nel DB. Come vedremo in seguito, il loro scambio è gestito da una classe che con metodi specifici, esegue codice SQL e ritorna, al modulo che ha invocato il metodo, i dati sotto forma di *Java Vector*.

Altre informazioni che possono essere scambiate, sono le funzioni per le quali viene invocato un servlet, il nome dell’utente, il codice della notizia su cui si vuole lavorare ecc. ecc.

A seconda della visibilità, si delineano tre livelli in cui i dati possono essere trattati:

?? Context
?? Session
?? Request

I dati che devono essere comuni fra tutti gli utenti sono scambiati a livello di context, cioè nel contesto dell’applicazione.

Maggiore attenzione meritano i dati relativi al singolo utente che accede al sistema. Per memorizzare e passare questo tipo di dato fra un’entità e l’altra del sistema, mantenendo l’individualità dell’utente, si usa l’oggetto *httpsession*. Ogni utente che si connette al sistema possiede una sessione univoca, che contiene informazioni al quale nessun altro utente può accedere. La vita di una determinata sessione viene interrotta quando l’utente si disconnette dal sistema o dopo un certo numero di minuti di inattività.

In questo modo, informazioni relative all’utente collegato che vengono spesso utilizzate durante la sua interazione con l’applicazione, possono venire memorizzate in sessione.

Nel motore di news, durante la fase di login viene memorizzato in sessione lo username dell’utente, in modo da poterlo identificare comodamente nelle sue richieste. Così per ogni azione di pubblicazione, creazione, inserimento nuovo utente e visualizzazione dei menù, si ha immediatamente a disposizione il dato relativo allo username dell’utilizzatore del programma, senza che si renda necessaria nessuna interrogazione al database.

L'oggetto session permette inoltre di superare la caratteristica del protocollo HTTP di essere *stateless*, cioè di non tenere traccia dei gesti precedentemente compiuti dall'utente. Infatti, la sessione può essere utilizzata per tenere una traccia del percorso compiuto dal suo proprietario durante il tempo in cui ha interagito con il sistema.

Questo tipo di astrazione consente di dare l'impressione all'utente che la sua interazione sia un unico susseguirsi di diverse azioni che fanno passare il sistema da uno stato all'altro, un po' come se si stesse utilizzando una connessione FTP.

Durante l'utilizzo dell'applicazione risulta molto comodo utilizzare la session per il passaggio dei dati fra un modulo e l'altro, in virtù del fatto che può essere considerato un oggetto condiviso, da tutto il sistema, relativamente al singolo utente utilizzatore.

Il terzo contesto in cui si possono scambiare i dati è la singola transizione. Abbiamo già parlato del paradigma *request-response* del protocollo HTTP. Per scambiare i dati relativi alla singola transizione si utilizza l'oggetto *HttpRequest* che viene usato come tramite, includendo gli oggetti che si vogliono passare. Java mette a disposizioni dei metodi della classe HttpRequest per compiere tutte le operazioni di inserimento e lettura dei dati nella richiesta.

## 4.2 La struttura modulare

Come anticipato nel capitolo della modellazione, l'applicazione è stata suddivisa in moduli, ognuno dei quali si occupa delle funzioni di uno stesso ambito. Questa organizzazione suddivide ogni tier[7] in ulteriori moduli, con compiti sempre più specializzati, rendendo più facili le operazioni di modica e di manutenzione del sistema.

[7] Livello della struttura Model-View-Controller

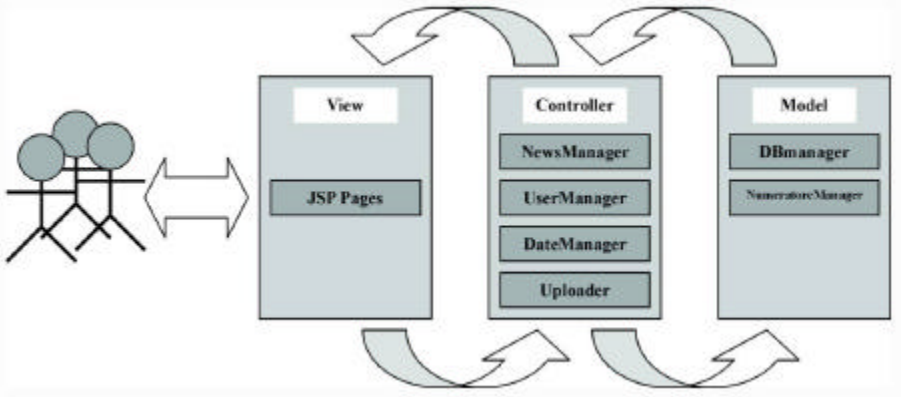


**Figura 35: Suddivisione in moduli dei tiers dell'applicazione**

In piccole applicazioni, molto spesso, la parte che svolge la funzione di *controller* è implementata con un singolo Java Servlet con il compito di smistare le richieste ad altre classi. Nel motore di news è stato scelto di spezzare il *controller* in più moduli che trattano le richieste di uno stesso tipo.

L’utente interagisce con il sistema in una sorta di ping-pong, nel quale per ogni funzione richiesta, viene mandata in risposta una pagina JSP.

Nei paragrafi che seguono saranno presentati uno per uno i moduli e le classi funzionali di cui è composto il motore, riportando alcuni esempi di codice.

### 4.2.1 Il modulo DBmanager

Il *Dbmanager* è una classe *Java* che estende la classe *HttpServlet* e si occupa sostanzialmente dell’esecuzione di statements SQL sui dati contenuti nel database. Facendo riferimento al pattern strutturale utilizzato, questo modulo si colloca all’interno del livello *Model*. Anche se sostanzialmente è un servlet, questa classe non produce codice HTML, ma si limita a reperire i dati dal DB tramite l’invocazione di metodi che eseguono operazioni di selezione, aggiornamento ed inserimento di record nelle tabelle.

Il Dbmanager con i suoi metodi è l’unica classe di tutta l’applicazione che si interfaccia alla sorgente di dati, qualsiasi essa sia. Rimane quindi aperta la possibilità di reperire i dati non da database, ma da qualsiasi altra

sorgente, come file di testo o altre applicazioni, modificando unicamente il Dbmanager.

I metodi che agiscono sui dati svolgono cinque tipi differenti di funzioni:

?? Selezione
?? Aggiornamento
?? Inserimento
?? Cancellazione
?? Controllo autorizzazioni

Ogni volta che viene chiamato un metodo del Dbmanager, vengono effettuate alcune operazioni in una sequenza predefinita che si ripete in tutti i metodi:

1. Apertura connessione verso il database
2. Esecuzione dello statement SQL
3. Chiusura della connessione

Questo modo di lavorare permette di tenere occupata la risorsa connessione per il minor tempo possibile in modo ottimizzarne l'utilizzo.

Tutto l'SQL eseguito dai metodi di Dbmanager è standard, e può essere eseguito su tutti i DBMS in commercio.

Riportiamo a titolo di esempio un esempio di codice Java di un metodo della classe Dbmanager per eseguire l'inserimento di un oggetto della classe News all'interno della base di dati.

**Metodo dbInsertNews della classe Dbmanager**

```
public static void dbInsertNews(News n) {

       int idNews = n.getIdNews();
       String cre = n.getCreatore();
       String tit = n.getTitolo();
       String cor = n.getCorpo();
       String firma = n.getFirma();
       int cancellata=0;
       int pubblicata=0;
       long data = n.getData();
       int tipo = n.getTipo();
       Connection con = null;

       try{
              //Carico il file di Classe del Driver
```

```
                Class.forName(driver);
                //Creo una connessione alla Base di Dati (db1)
                con = DriverManager.getConnection(connessione,"","");

                        //Creo l'oggetto statement
                Statement statement = con.createStatement();
                statement.executeUpdate("INSERT INTO News
(IDnews,creatore,titolo,corpo,firma,data,cancellata,pubblicata,tipo)
VALUES ("+idNews +",'"+cre+"' ,'"+tit+"' ,'"+cor+"','"+firma+"',
"+data+","+cancellata+","+pubblicata+","+tipo+")");

        }
        catch (SQLException sqle) {
                System.err.println(sqle.getMessage());
        }
        catch (ClassNotFoundException cnfe) {
                System.err.println(cnfe.getMessage());
        }
        catch (Exception e) {
                System.err.println(e.getMessage());
        }
        finally {
                try {
                        if (con!=null) {
                                con.close();
                        }
                }
                catch (SQLException sqle) {
                        System.err.println(sqle.getMessage());
                }
        }
        }
```

**Figura 36: Esempio di codice java del metodo dbInsertNews**

La tabella mostra i metodi della classe Dbmanager, separati per tipo di funzioni svolte:

| | |
|---|---|
| dbSelect<br>dbSelectBoolean<br>dbSelectInteger<br>dbSelectIntString | Selezione |
| dbPubblica<br>dbSetString | Aggiornamento |
| dbDelete | cancellazione |
| dbInsertFunzione<br>dbInsertImmagine<br>dbInsertNews<br>dbinsertLink | Inserimento |

| | |
|---|---|
| dbInsertUtente | |
| dbAuthorization<br>dbLogin | Controllo autorizzazioni |

**Figura 37: Metodi della classe Dbmanager**

Il listato completo dei metodi della classe Dbmanger è mostrato nella sezione relativa al codice.

La differenza fra i vari metodi che svolgono lo stesso tipo di funzione, è dovuta al diverso formato dei dati trattati e all’esigenza di scrivere query differenti a seconda del tipo di dato sul quale vuole agire.

### 4.2.2 L’interfaccia RecordPrototype

Come già mostrato nel class diagram, gli oggetti istanziati che rappresentano i dati memorizzati nel database appartengono a cinque classi:

- News
- Utente
- Funzione
- Link
- Immagine

La struttura delle tabelle del database ricalca esattamente il prototipo di queste classi, in modo che esista una corrispondenza diretta fra i record di una tabella e gli attributi della corrispondente classe Java.

Ogni volta che si effettua una query di selezione è necessario che il metodo che esegue tale operazione sia a conoscenza della struttura dei record che otterrà in risposta dal database, rendendo necessaria la costruzione di almeno un metodo diverso per ogni tabella su cui si effettuano operazioni di selezione di record.

Questo inconveniente è stato evitato con l’uso dell’interfaccia RecordPrototype. Le classi News, Utente, Funzione, Link e Immagine implementano questa interfaccia, ovvero hanno un comportamento comune che le rende soggette ad operazioni comuni. Queste operazioni sono rappresentate dai metodi dell’interfaccia, i quali possono essere invocati per ognuna delle classi che implementa l’interfaccia.

I metodi di RecordPrototype sono:

?? getInstance
?? setValues

Il primo non fa altro che restituire un’istanza della classe di cui fa parte l’oggetto che invoca il metodo, mentre il secondo ‘riempie’ gli attributi dell’istanza.

RecordPrototype

```
package news;

public interface RecordPrototype {

public RecordPrototype getInstance();
public void setValues(java.sql.ResultSet rs);

}
```

**Figura 38: codice dell'interfaccia RecordPrototype**

L’importanza di questo modo di lavorare, sta nel fatto che nell’invocazione di un metodo di selezione, si passa, fra i parametri del metodo stesso, un’istanza dell’oggetto contenuto nella tabella. Di questa istanza non si fornisce il tipo, ma l’interfaccia, in modo che all’interno del codice del metodo, sia possibile maneggiare l’istanza senza sapere di che classe fa parte, ma utilizzando i due metodi getInstance e setValues.

Quindi all’interno del metodo di selezione si provvederà a istanziare la classe passata (getInstance) per poi riempire l’istanza con i valori estratti dalla tabella (setValues).

In questo modo abbiamo standardizzato un grande numero di metodi di selezione che, diversamente, avrebbero dovuto essere scritti ‘su misura’ al formato dei record selezionati.

In figura è mostrato il codice di un metodo di selezione che utilizza questo meccanismo.

Metodo dbSelect della classe DBmanager

```
public static Vector dbSelect(RecordPrototypeprototipo, String tabella) {

       Connection con = null;
       //Istanzio il vettore che uso da Bean
       Vector dbb = new Vector();
       try{
              //Carico il file di Classe del Driver
              Class.frName(driver);
              //Creo una connessione alla Base di Dati utilizzando l'ODBC
(db1)
              con = DriverManager.getConnection(connessione,"","");

              //Creo l'oggetto statement
              Statement statement = con.createStatement();

              //Eseguo una Query di selezione generica e metto il
risultato in rs
              ResultSet rs = statement.executeQuery("SELECT * FROM
"+tabella);
               while (rs.next()) {
                     RecordPrototype prot = prototipo.getInstance();
                     prot.setValues(rs);
                     dbb.add(prot);
               }
              rs.close();
       }
       catch (SQLException sqle) {
              System.err.println(sqle.getMessage());
              System.err.println("Problema di SQL: è stata scatenata
un'eccezione");
       }

       catch (ClassNotFoundException cnfe) {
              System.err.println(cnfe.getMessage());
       }
       catch (Exception e) {
              System.err.println(e.getMessage());
       }
       finally {

              try {
                     if (con!=null) {
                            con.close();
                     }
              }
              catch (SQLException sqle) {
                     System.err.println(sqle.getMessage());
                     System.err.println("Problema SQL: è stata scatenata
un'eccezione nella clausola finally del metodo dbselect");
              }
       }
       return dbb;
}
```

**Figura 39: Listato del codice del metodo Dbselect**

In neretto sono indicate le righe di codice in cui vengono effettuate le operazioni descritte.

In particolare, tali operazioni, vengono effettuate all'interno di un ciclo *While* durante l'analisi del ResultSet[8]. Quindi per ogni record trovato, viene istanziato un oggetto della classe (senza sapere quale essa sia) e viene 'riempito' con in dati del record le cui colonne corrisponderanno con gli attributi dell'istanza della classe.

Nell'utilizzare RecordPrototype è stata applicata in pieno la filosofia della programmazione ad oggetti, trattando un'istanza di una classe conoscendo il suo comportamento generale detto appunto *interfaccia*.

### 4.2.3 La classe NumeratoreManager

Come già anticipato nel secondo capitolo, parlando dei problemi di portabilità, è utile implementare una funzione che restituisca automaticamente, ad ogni news inserita, un intero che verrà utilizzato come identificatore univoco all'interno del sistema.

La classe NumeratoreManager fa parte del tier *Model* e si occupa, con il suo metodo *getNextNumber*, di ottenere un identificatore univoco della news nel sistema.

Il metodo *getNextNumber* si occupa di accedere all'unico record della tabella *Contatori* e, dopo averne letto il valore, incrementarlo di una unità.

L'accesso a tale dato deve essere regolato in modo che rispetti le regole di consistenza nel caso di accesso concorrente di differenti processi.

Questo si ottiene dichiarando il metodo di tipo *synchronized*.

Dedicare una tabella del DBMS per ospitare un solo record, può sembrare una scelta bizzarra, ma è, a nostro parere, il modo più corretto per implementare questa funzione.

NumeratoreManager

```
package news;

import java.sql.*;

public class NumeratoreManager {

private NumeratoreManager() {
       super();
}
```

[8] In Java per ResultSet s'intende l'insieme dei dati restituiti dalla query di interrogazione effettuata su una tabella della base dei dati.

```
public static synchronized int getNextNumber(int idNumeratore) {
Connection con = null;
int nextValue = 0;
	try {
		//apro connessione con database

		//Carico il file di Classe del Driver
		Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

		//Creo una connessione alla Base di Dati utilizzando l'ODBC
(db1)
		con =
DriverManager.getConnection("jdbc:odbc:DataBaseNews","","");

		//Creo l'oggetto statement
		Statement statement = con.createStatement();

		ResultSet rs = statement.executeQuery("select * from
Contatori where id=" + idNumeratore);
		while (rs.next()) {
			nextValue = rs.getInt("value");
			nextValue++;
		}
		statement.executeUpdate("UPDATE Contatori SET
value="+nextValue+" WHERE id="+idNumeratore);


	}
	catch (SQLException sqle) {

			System.err.println(sqle.getMessage());
		}
	catch (Exception ex) {
		ex.printStackTrace();
	}
	finally {
		try {
			if (con!=null) {
				con.close();
			}
		}
		catch (SQLException sqle) {

			System.err.println(sqle.getMessage());
		}
	}
	return nextValue;
}
}
```

**Figura 40: listato del codice della classe NumeratoreManager**

### 4.2.4 Il modulo UserManager

L’UserManager è il cuore di tutte le funzioni riguardanti gli utenti. Estende la classe *HttpServlet* e, facendo riferimento al pattern strutturale MVC, questo modulo si colloca all’interno del livello *Controller*.

Ad esso giungono le chiamate alle funzioni offerte dall’interfaccia utente e si occupa di effettuare le operazioni di:

- Login
- Visualizzazione menù Super User
- Visualizzazione Modulo per l’inserimento di un nuovo utente
- Inserimento di un nuovo Utente
- Visualizzazione elenco utenti
- Logout

Il tipo di funzione per cui è stato invocato viene passato come parametro nella request sotto il nome di function.

A seconda della funzione selezionata verranno eseguiti i frammenti di codice che compiono le seguenti operazioni:

- Controllo autorizzazioni
- Interazione con il tier *Model*
- Elaborazione dei dati
- Passaggio dei dati e del controllo all’opportuna pagina JSP

Lo UserManager è quindi suddiviso in tanti blocchi quante sono le possibili funzioni.

A titolo di esempio riportiamo il codice relativo alla funzione di inserimento di un nuovo utente:

funzione inserimento della classe User Manager

```
if (function.equals("inserisci_utente")){
                        String username_in_session =
(String)session.getValue("username");
                        //Istanzio un oggetto della classe Funzione per
passare il prototipo
                        Funzione new_f = new Funzione();
                        //controllo delle autorizzazioni
                        //eseguo la query e carico il risultato nel bean
                        Vector dbb =
DBmanager.dbSelect(new_f,"Funzioni","username",username_in_session);
                        boolean ok = DBmanager.dbAuthorization(dbb,"auto");
                        if (ok){
                                //assegno i dati a variabili locali
                                String new_username =
request.getParameter("new_username");
                                String new_password =
request.getParameter("new_password");
                                String new_firma =
request.getParameter("new_firma");
                                int new_scrivi=0;
                                int new_pubblica=0;
                                int new_autorizza=0;
                                String cb =
(String)request.getParameter("scrivi");
                                if (cb.equals("on")){
                                        new_scrivi=1;
                                }
                                cb =
(String)request.getParameter("pubblica");
                                if (cb.equals("on")){
                                        new_pubblica=1;
                                }
                                cb =
(String)request.getParameter("autorizza");
                                if (cb.equals("on")){
                                        new_autorizza=1;
                                }
                                Utente u = new Utente();
                                dbb =
DBmanager.dbSelect(u,"Utenti","username",new_username);
                                //se ci sono altri utenti con lo stesso
username...errore
                                if(!dbb.isEmpty()){
       getServletConfig().getServletContext().getRequestDispatcher("/delfo
/news/username_non_valido.jsp").forward(request, response);
                                }
                                else{
                                        u.setUsername(new_username);
                                        u.setPassword(new_password);
                                        u.setFirma(new_firma);
                                        //inserisco l'utente nella tabella
utenti
                                        DBmanager.dbInsertUtente(u);
                                        new_f.setUsername(new_username);
                                        new_f.setScrivi(new_scrivi);
```

```
                                    new_f.setPubblica(new_pubblica);
                                    new_f.setAutorizza(new_autorizza);
                                    //inserisco la funzione nella tabella

                                    DBmanager.dbInsertFunzione(new_f);
      getServletConfig().getServletContext().getRequestDispatcher("/delfo
/news/conferma_utente.jsp").forward(request, response);
                                }
                        }
                        else{
      getServletConfig().getServletContext().getRequestDispatcher("/delfo
/news/operazione_non_permessa.jsp").forward(request, response);
                        }
                }
```

**Figura 41: funzione di inserimento di un nuovo utente nel sistema**

## *Il controllo delle autorizzazioni*

Si è già parlato del metodo della classe Dbmanager che controlla le autorizzazioni, ma rimane da affrontare un problema concettuale legato a tale operazione. Il problema è decidere in quali occasioni sia necessario effettuare il controllo.

Sicuramente è indispensabile accertarsi che l’utente sia in possesso dei diritti necessari, quando si accede ai dati nel database, sia in scrittura, che in aggiornamento o in lettura. Questo per evitare che utenti non autorizzati possano inserire, modificare o solamente leggere dati al quale non sono autorizzati ad accedere. Utilizzando la struttura a menù personalizzati che abbiamo implementato, si rende necessario il controllo anche prima di presentare la lista delle funzioni all’utente. Questa scelta deriva dall’esigenza di nascondere all’utente le funzioni ad egli vietate per evitare un uso improprio del sistema e per non appesantire l’applicazione con l’onere di informare l’utente dell’impossibilità di accedere alla funzione selezionata.

Questo si traduce in un controllo delle autorizzazione ad ogni gesto e ad ogni interazione con l’interfaccia grafica.

L’operazione di controllo comporta un impegno di risorse da parte del server e, nel caso di un accesso concorrente di utenti, può rappresentare una variabile sulla quale dimensionare l’hardware del sistema.

Si aprono due soluzioni, entrambe valide, ma caratterizzate da punti a sfavore che costringono ad un’attenta analisi del problema prima di scegliere l’una rispetto all’altra.

Della prima soluzione abbiamo appena parlato, consiste cioè nell'effettuare il controllo ad ogni gesto dell'utente, in modo da tenere sotto controllo in ogni momento l'ambiente con il quale egli interagisce.

Le chiamate alle funzioni dei servlet sono effettuate, in molti casi, con il metodo GET del protocollo HTTP e, anche se la funzione vietata non viene offerta dal menù, è facile eludere questo tipo di controllo immettendo direttamente la stringa giusta, nella barra degli indirizzi del browser utilizzato. Naturalmente anche in questo caso il sistema deve effettuare i controlli prima di scatenare l'esecuzione della funzione richiesta. Per questi motivi siamo in presenza di un alto numero di controlli di autorizzazioni che si traducono in quattro fasi elementari:

1. Apertura di connessione verso il database
2. Selezione del record della tabella *Funzioni* relativo all'utente che ha effettuato il login
3. Elaborazione e controllo dei dati ottenuti
4. Chiusura della connessione

L'alta frequenza con la quale questi passi vengono ripetuti, rappresenta il punto a sfavore nell'utilizzo di questa tecnica in sistemi molto frequentati, ma è sicuramente il modo migliore per evitare violazioni della sicurezza.

In figura è mostrato il codice del metodo dbAuthoritazion del metodo Dbmanager.

Metodo dbAuthorization del modulo DBmanager

```
public static boolean dbAuthorization(Vector v, String
request_authorization) {

       boolean permission = false;
       for(int i=0;i<v.size();i++){
              Funzione f = (Funzione)v.elementAt(i);

              if (request_authorization.equals("write")){
                            if (f.isScrivi() == 1) {
                            permission=true;
                     }
              }

              if(request_authorization.equals("pubb")){
                            if (f.isPubblica() == 1) {
                            permission=true;
                     }
              }

              if(request_authorization.equals("auto")){
```

```
                                if (f.isAutorizza() == 1) {
                                permission=true;
                        }
                }
        }

        return permission;

}
```

**Figura 42: Codice del metodo dbAuthorization per il controllo delle autorizzazioni**

Il secondo metodo è assolutamente identico al primo per quello che riguarda la frequenza dei controlli, ma si differenzia per il modo con il quale questi vengono effettuati.

Infatti, invece di accedere ogni volta alla base dei dati, è sufficiente inserire nell’oggetto *httpsession* del singolo utente la descrizione dei suoi diritti. Questa operazione, che viene effettuata all’atto del login, permette di accedere ad una fonte dei dati che non è più il database, ma è un oggetto in memoria. Il peso del controllo è inferiore, ma, a livello teorico, la procedura utilizzata è imperfetta.

Immaginiamo che un utente in possesso dei diritti necessari, stia utilizzando il sistema in modo errato, causando problemi agli altri utenti che segnalano in tempo reale l’inconveniente all’amministratore del sistema, il quale provvede a rimuovere immediatamente l’utente dalle tabelle, in modo da togliergli ogni tipo di diritto. Il problema è che il sistema riconoscerebbe all’utente indisciplinato i diritti di cui era in possesso in fase di login, cioè quando sono stati copiati in sessione. Questa caratteristica rende inutile la revoca delle autorizzazioni effettuate dal superuser nel momento in cui l’utilizzatore è connesso al sistema.

Nella realizzazione del prototipo dell’applicazione è stato utilizzato il primo metodo, perché ritenuto più esatto di quanto non lo sia il secondo. È comunque vero che in presenza di sistemi molto frequentati, l’uso della seconda soluzione deve essere attentamente considerato.

### 4.2.5 Il modulo NewsManager

Il NewsManager estende la classe *httpServlet* e si occupa di gestire tutte le funzioni che agiscono sulle news. Dalle pagine Jsp dell’interfaccia utente, vengono invocate con il metodo GET del protocollo http, le seguenti funzioni del NewsManager come valore dell’attributo *function_news*:

1. Visualizzazione lista delle news pubblicate
2. Lettura della singola news pubblicata
3. Visualizzazione delle news scritte dall’utente
4. Cancellazione delle news scritte dall’utente
5. Visualizzazione di tutte le news (ancora da pubblicare e pubblicate)
6. Cancellazione di una news
7. Pubblicazione di una news
8. Visualizzazione del form d’inserimento di nuove news
9. Inserimento di nuove news
10. Upload di immagine allegata alla news
11. Recupero di news precedentemente cancellate

All’atto della chiamata a questo servlet, viene specificata, come parametro, la funzione da svolgere. Il NewsManager eseguirà il frammento di codice preposto a tale funzione.

Per ogni funzione selezionata vengono eseguite in sequenza le seguenti operazioni:

?? Controllo autorizzazioni
?? Interazione con il tier *Model*
?? Elaborazione dei dati
?? Passaggio dei dati e del controllo all’opportuna pagina JSP

Come indicato, ogni funzione termina con il passaggio del controllo ad una pagina JSP. Questo permette all’utente di ottenere una risposta per ogni funzione selezionata.

### 4.2.6 La classe NewsBean

I dati relativi alle news vengono spesso passati fra servlet (NewsManager) e pagine JSP. Come ‘veicolo’ di tale passaggio viene utilizzata una classe apposita: il NewsBean.

La classe NewsBean è un oggetto in grado di contenere un’istanza della classe *News*, un numero indefinito di istanze di *Link* e un’istanza di *Immagine.*

In questo modo, vengono passati tutti i dati relativi alle news, incapsulati in un unico oggetto. Alle pagine Jsp che mostrano liste di notizie, la classe NewsManager passa una *Enumeration* di NewsBean.

NewsBean

```
package news;

import java.util.Vector;

/**
 * Insert the type's description here.
 * Creation date: (24/04/2001 17.06.07)
 * @author: Administrator
 */
public class NewsBean {

	private News news;
	private Vector vlink;
	private Vector vimmagine;


public NewsBean() {
	super();
}

public News getNews() {
	return news;
}

public java.util.Vector getVimmagine() {
	return vimmagine;
}

public java.util.Vector getVlink() {
	return vlink;
}

public void setNews(News newNews) {
	news = newNews;
}

public void setVimmagine(java.util.Vector newVimmagine) {
	vimmagine = newVimmagine;
}
```

```
public void setVlink(java.util.Vector newVlink) {
       vlink = newVlink;
}
}
```

**Figura 43: listato del codice della classe NewsBean**

Riportiamo a titolo di esempio un frammento di codice della funzione di lettura delle News, con evidenziate le righe in cui viene costruita l'Enumeration di NewsBean.

Funzione di lettura delle news pubblicate di NewsManager

```
if (function.equals("leggi")) {
                      //istanzio il DBmanager per eseguire la query
                      //DBmanager dbman = new DBmanager();
                      //Istanzio un oggetto della classe News per passare
il prototipo
                      News n = new News();
                      //eseguo la query e carico il risultato nel bean
                      //Vector dbb =
dbman.dbSelectBoolean(n,"News","pubblicata",true,"cancellata",false);
                      Vector dbb =
DBmanager.dbSelectBoolean(n,"News","pubblicata",1,"cancellata",0);

                      Vector vector_bean = new Vector();

                      //per ogni news trovata, costruisco una enumeration
di NewsBean
                      for (int i=0;i<dbb.size();i++){
                             //istanzio un nuovo NewsBean
                             NewsBean nb = new NewsBean();
                             News news = (News)dbb.elementAt(i);
                             //inserisco la News nel News Bean
                             nb.setNews(news);
                             //Istanzio un oggetto della classe Link per
passare il prototipo
                             Link l = new Link();
                             Vector dbb_link =
DBmanager.dbSelectInteger(l,"Links","ID",news.getIdNews());
                             //inserisco il vettore dei link nel Newsbean
                             nb.setVlink(dbb_link);
                             Immagine img = new Immagine();
                             Vector dbb_img =
DBmanager.dbSelectInteger(img,"Immagini","ID",news.getIdNews());
                             nb.setVimmagine(dbb_img);
                             //costruisco il vettore dei Newsbean
                             vector_bean.addElement(nb);
                      }
                      Enumeration enum = vector_bean.elements();
                      //inserisco il bean nella richiesta
                      request.setAttribute("EnumBean",enum);

                      //passo il controllo alla pagina JSI
```

```
       getServletConfig().getServletContext().getRequestDispatcher("/delfo
/news/leggi_news.jsp").forward(request, response);
              }
```

**Figura 44: Esempio di codice in cui viene passato un oggetto Enumeration alla JSP**

Prima vengono costruiti i singoli NewsBean e aggiunti ad un *Vector*. Alla fine di tutto il vettore viene riversato interamente in una *Enumeration*.

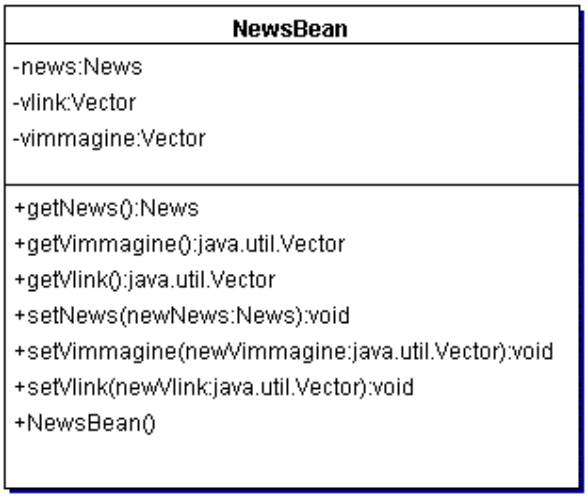


**Figura 45: Descrizione UML della calsse NewsBean**

I link e le immagini sono contenuti all’interno del NewsBean sottoforma di vettori, mentre la notizia è contenuta in un oggetto di tipo News. I metodi danno la possibilità di accedere dall’esterno a tali attributi, sia in lettura che in scrittura.

### 4.2.7 La classe DateManager

Nei sistemi in cui compaiono degli oggetti che rappresentano una data, sorgono problemi sotto l’aspetto della conversione fra i vari formati in cui si può rappresentare una data.

Nel nostro caso il problema era quello di visualizzare sottoforma di stringa di caratteri un oggetto istanza della classe *Date*.

A tale proposito si è reso necessario costruire una apposita classe che attraverso il suo metodo *haveDate*, trasforma un oggetto data nella sua rappresentazione.

DateManager

```
package news;

import java.util.*;

public class DateManager {

private DateManager() {
       super();
}

public static String haveDate(Date data) {

       String data_str;

       GregorianCalendar cal = new GregorianCalendar();
       cal.setTime(data);
       data_str = cal.get(Calendar.DAY_OF_MONTH) + "/"
+(cal.get(Calendar.MONTH)+1) + "/" +
                        cal.get(Calendar.YEAR);

       return data_str;

}
}
```

**Figura 46: listato del codice della classe DateManager**

### 4.2.8 La classe Uploader

La classe Uploader si occupa di gestire l'upload di un file JPG dal computer remoto al server. Come NewsManager e UserManager, anche questa classe estende la classe *httpServlet*.

All'atto dell'upload del file, viene creata nel file System una directory provvisoria con il nome dell'identificativo di sessione dell'utente che sta utilizzando il sistema. Inseguito il NewsManager provvederà ad effettuare un ridimensionamento dell'immagine e spostarla in una directory utente. La cartella provvisoria viene cancellata al termine dell'operazione.

Il dimensionamento dell'immagine si rende necessario per standardizzare la dimensione su disco delle immagini caricate dagli utenti creatori di news. Infatti nell'eventualità che venga allegato ad una news un file di dimensioni troppo elevate, si avrebbero problemi sia in

visualizzazione per l’occupazione di una porzione di browser troppo ampia, sia in download dell’immagine sui client che leggono la notizia.

L’operazione di ridimensionamento avviene tramite un metodo di trasformazioni affini, che garantisce il mantenimento delle proporzioni ed una corretta visualizzazione dell’immagine.

Uploader

```
package news;

import com.oreilly.servlet.multipart.*;

import test.upload.FileContainer;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;


public class Uploader extends javax.servlet.http.HttpServlet {

public void doGet(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws
javax.servlet.ServletException, java.io.IOException {

       performTask(request, response);

}

public void doPost(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse response) throws
javax.servlet.ServletException, java.io.IOException {

       performTask(request, response);

}

public String getServletInfo() {

       return super.getServletInfo();

}

public void init() {
       // insert code to initialize the servlet here

}

public void performTask(javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse res) {

       try
       {
              HttpSession session = request.getSession();
              FileContainer dir = new FileContainer(session.getId());
              session.putValue("FILE_CONTAINER", dir);


              MultipartParser multiReq = new MultipartParser(request,
```

```
1000000,true,true);
               Part part = null;
               while (true) {
                      part = multiReq.readNextPart();
                      if (part.isFile()) {
                             break;
                      }
               }
               String name = dir.addFile((FilePart) part);
               System.out.println("Uploadato file: "+name);

       getServletConfig().getServletContext().getRequestDispatcher("/delfo
/news/conferma_inserimento_immagine.jsp").forward(request, res);
               }

       catch(Throwable theException)
       {
               theException.printStackTrace();
               // uncomment the following line when unexpected exceptions
               // are occuring to aid in debugging the problem.
               //theException.printStackTrace();
       }
}
}
```

**Figura 47: listato della classe Uploader**

### 4.2.9 La classe HTMLEncoder

Questa classe risolve i problemi che sorgono durante la fase di inserimento di dati nel sistema. Tali malfunzionamenti sono dovuti alla necessità di inserire, nel titolo o nel testo delle notizie, particolari caratteri che disturbano per diversi motivi il corretto funzionamento del motore di news.

La classe *HTMLEncoder*, con il suo metodo *convertString*, permette di prendere in ingresso una stringa di caratteri e convertirla in un'altra stringa in cui alcuni caratteri sono stati sostituiti dalla codifica degli stessi caratteri definita dallo standard HTML.

Presentiamo una rassegna dei caratteri problematici.

#### *Doppio apice (")*

Il doppio apice viene utilizzato spesso per delimitare variabili di tipo stringa e, in molti linguaggi di programmazione è riservato a particolari variabili. Per motivi di sicurezza, il doppio apice viene convertito dall'HTMLENcoder nella sequenza indicata fra parentesi: (").

### *Carattere di andata a capo (/n)*

Nella visualizzazione di un testo in una pagina HTML, l'andata a capo deve essere segnalata con un tag specifico (<BR>). Questo rende necessario sostituire con tale tag le sequenze di andata a capo che compaiono nel testo inserito da tastiera (/n). Diversamente, il testo verrebbe rappresentato su una unica riga, stravolgendo l'impaginazione data dall'utente creatore della notizia.

### *Singolo apice (')*

Tutte le volte che si utilizza il singolo apice utilizzato come apostrofo o accento, l'inserimento dei dati nel database risulterebbe errato a causa dell'interpretazione del carattere (') come delimitatore stringa all'interno delle istruzioni SQL. Questo problema si risolve sostituendo ogni singolo apice con la sequenza (').

### *Simbolo di maggiore (>)*

Il simbolo di maggiore viene interpretato dal browser come la chiusura di un tag HTML, stravolgendo la visualizzazione del testo che segue tale carattere.

Il problema si risolve esaminando le stringhe inserite e sostituendo tale simbolo con la sequenza (>).

### *Simbolo di minore (<)*

Analogamente al caso precedente, il simbolo di minore viene interpretato dal browser come l'apertura di un tag HTML.

La sequenza da sostituire è, in questo caso, (<).

Utilizzando la classe HTMLEncoder abbiamo protetto il motore di news dagli inconvenienti dovuti all'introduzione nel sistema da caratteri speciali e alla loro visualizzazione in pagine HTML. Tuttavia non è escluso che l'installazione dell'applicazione in particolari situazioni richieda un controllo e una sostituzione di altri caratteri. Tale operazione richiederebbe una banale modifica del metodo convertString di questa classe.

Metodo convertString della classe HTMLEncoder

```
public static String convertString(String oldStr) {
  if ((oldStr != null) && (oldStr.length() != 0)) {
       StringBuffer newStr = new StringBuffer();
         for (int i = 0; i < oldStr.length(); i++) {
                try {
                  if (oldStr.charAt(i) == '"')
                        newStr.append("&quot;");
                      else
                        if (oldStr.charAt(i) == '\n')
                                    newStr.append("<BR>");
                        else
                          if (oldStr.charAt(i) == '\r')
                                         newStr.append("");
                          else
                            if (oldStr.charAt(i) == '\'')

       newStr.append("&#039;");
                             else
                               if (oldStr.charAt(i) == '>')

       newStr.append("&gt;");
                                else
                                   if (oldStr.charAt(i) == '<')

       newStr.append("&lt;");
                                   else


       newStr.append(oldStr.charAt(i));
                    catch (Exception ex) {
                             System.out.println("StringError:" + ex);
                             ex.printStackTrace();
                    }
       }
              return newStr.toString();
  }
             return oldStr;
}
```

**Figura 48: codice del metodo che effettua la conversione delle stringhe**

### 4.2.10 Le pagine Jsp

Tutta la parte relativa al tier *View* è composta da parte JSP. Queste pagine rappresentano un'estensione della tecnologia Java Servlet e permettono di avere un maggior controllo e una maggiore leggibilità dell'HTML che viene inviato al client.

La prima volta che viene eseguita una pagina JSP, il compilatore Java converte la pagina in un Servlet equivalente. Dal punto di vista della JVM[9] è perfettamente uguale interpretare un Java Servlet o una Java Servlet Page.

Riportiamo ora il codice di una semplice pagina JSP che visualizza il menù iniziale e il servlet generato.

home.jsp

```
<%@ page errorPage="errorpage.jsp" %>
<%@ page language ="java" %>
<%@ page import ="news.DBmanager" %>
<%@ page import ="java.util.Vector" %>
<%@ page import ="news.Funzione" %>

<html>
<head>
<META HTTP-EQUIV="Expires" CONTENT="Wed, 16 June 1999 00:00:00 GMT">
<META HTTP-EQUIV="pragma" CONTENT="no-cache">
<title>Home page</title>
</head>

 <body bgcolor="#FFFFFF">
 <%
       //la variabile locale alla pagina username contiene lo username in
sessione
       String username=(String)session.getValue("username");
       //vado a prendere dalla session il valore degli attributi in
sessione
       Funzione fun = (Funzione)session.getValue("diritti");
 %>
   <font face="verdana">
   <font color="navy"><b><p align="center">Ciao <%=username %>, benvenuto
nel motore di News</b></p></font>
   <table align ="center" border ="0" >
   <tr>
   <td bgcolor="#CC9988" colspan="2"><b> </b>
   </td>
         <td>
    <img src="/delfo/News/images/lente.gif">
   </td>
   <td valign="middle">
    <b><a href="/servlet/news.NewsManager?function_news=leggi" alt="leggi
```

[9] Java Virtual Machine: l'interprete del Java Bytecode

```
news"><img src="/delfo/News/images/scritta_leggi.gif"
border="0"></a></b><br>
   </td>
   </tr>
        <%
        if((int)fun.isScrivi()==1){
        %>
                  <tr>
                  <td>
                  <img src="/delfo/News/images/matita.gif">
                  </td>
                  <td>
                  <b><a
href="/servlet/news.NewsManager?function_news=crea"><img
src="/delfo/News/images/scritta_nuova_news.gif" border="0"></a></b><br>
                  </td>
                  </tr>
                  <tr>
                  <td>
                  <img src="/delfo/News/images/sbagliate.gif">
                  </td>
                  <td>
                  <b><a
href="/servlet/news.NewsManager?function_news=modifica"><img
src="/delfo/News/images/proprie_news.gif" border="0"></a></b><br>
                  </td>
                  </tr>
        <%
        }
        %>
        <%
        if((int)fun.isPubblica()==1){
        %>
                  <tr>
                  <td>
                  <img src="/delfo/News/images/pubblica_grande.gif">
                  </td>
                  <td>
                  <b><a
href="/servlet/news.NewsManager?function_news=pubblica"><img
src="/delfo/News/images/scritta_pubblicazione.gif" border="0"></a></b><br>
                  </td>
                  </tr>
        <%
        }
        %>
        <%
        if((int)fun.isAutorizza()==1){
        %>
                  <tr>
                  <td>
                  <img src="/delfo/News/images/schedario.gif">
                  </td>
                  <td>
                  <b><a
href="/servlet/news.UserManager?function=superuser"><img
src="/delfo/News/images/scritta_autorizzazioni.gif"
border="0"></a></b><br>
                  </td>
                  </tr>
```

```
       <%
       }
       %>
   <tr>
   <td bgcolor="#CC9988" colspan="2"><b> </b>
   </td>
   <td>
   </td>
   </tr>
    <tr>
   <td colspan="2" align="center"><a
href="/servlet/news.UserManager?function=logout"><img
src="/delfo/News/images/scritta_logout.gif" border="0"></a><br></b>
   </td>
   <td>
   </td>
   </tr>
   </table>
  </font>
  </font>
 </body>
</html>
```

**Figura 49 : codice della pagina home.jsp**

Servlet generato dalla pagina home.jsp

```
public void _jspService(HttpServletRequest request, HttpServletResponse
response)
              throws IOException, ServletException {
              if (isHaltAtServiceBegin) {

       Environment/hosts/default_host/default_app/web/delfo/news/home.jsp
1,1
                     com.ibm.uvm.tools.DebugSupport.halt();
              }
              boolean _jspx_cleared_due_to_forward = false;
              JspFactory _jspxFactory = null;
              PageContext pageContext = null;
              HttpSession session = null;
              ServletContext application = null;
              ServletConfig config = null;
              JspWriter out = null;
              Object page = this;
              String  _value = null;
              try {
                     if (_jspx_inited == false) {
                            _jspx_init();
                            _jspx_inited = true;
                     }
                     _jspxFactory = JspFactory.getDefaultFactory();
                     try
                     {
                            response.setContentType("text/html");
                     }
                     catch (IllegalStateException ws_jsp_ise)
                     {
                     }
                     pageContext = _jspxFactory.getPageContext(this,
request, response,
```

```
                    "errorpage.jsp", true, 8192, true);
                    application = pageContext.getServletContext();
                    config = pageContext.getServletConfig();
                    session = pageContext.getSession();
                    out = pageContext.getOut();
                    out.print(new String(_jspx_html_data[0]));
                    out.print(new String(_jspx_html_data[1]));
                    out.print(new String(_jspx_html_data[2]));
                    out.print(new String(_jspx_html_data[3]));
                    out.print(new String(_jspx_html_data[4]));
                    out.print(new String(_jspx_html_data[5]));
                    // begin [file=C:/Programmi/IBM/VisualAge for
Java/ide/project_resources/IBM WebSphere Test
Environment/hosts/default_host/default_app/web/delfo/news/home.jsp;from=(1
6,3);to=(24,1)]
                                    //la variabile locale alla pagina
username contiene lo username in sessione
                                    String
username=(String)session.getValue("username");

                                    //vado a prendere dalla session il
valore degli attributi in sessione

                                    Funzione fun =
(Funzione)session.getValue("diritti");
                    // end
                    out.print(new String(_jspx_html_data[6]));
                    // begin [file=C:/Programmi/IBM/VisualAge for
Java/ide/project_resources/IBM WebSphere Test
Environment/hosts/default_host/default_app/web/delfo/news/home.jsp;from=(2
6,51);to=(26,60)]
                            out.print(username );
                    // end
                    out.print(new String(_jspx_html_data[7]));
                    // begin [file=C:/Programmi/IBM/VisualAge for
Java/ide/project_resources/IBM WebSphere Test
Environment/hosts/default_host/default_app/web/delfo/news/home.jsp;from=(5
0,6);to=(52,2)]
                                    if((int)fun.isScrivi()==1){
                    // end
                    out.print(new String(_jspx_html_data[8]));
                    // begin [file=C:/Programmi/IBM/VisualAge for
Java/ide/project_resources/IBM WebSphere Test
Environment/hosts/default_host/default_app/web/delfo/news/home.jsp;from=(7
0,4);to=(72,3)]
                                    }
                    // end
                    out.print(new String(_jspx_html_data[9]));
                    // begin [file=C:/Programmi/IBM/VisualAge for
Java/ide/project_resources/IBM WebSphere Test
Environment/hosts/default_host/default_app/web/delfo/news/home.jsp;from=(7
5,5);to=(77,2)]
                                    if((int)fun.isPubblica()==1){
                    // end
                    out.print(new String(_jspx_html_data[10]));
                    // begin [file=C:/Programmi/IBM/VisualAge for
Java/ide/project_resources/IBM WebSphere Test
Environment/hosts/default_host/default_app/web/delfo/news/home.jsp;from=(8
7,4);to=(89,3)]
                                    }
```

```
                    // end
                    out.print(new String(_jspx_html_data[11]));
                    // begin [file=C:/Programmi/IBM/VisualAge for
Java/ide/project_resources/IBM WebSphere Test
Environment/hosts/default_host/default_app/web/delfo/news/home.jsp;from=(9
2,4);to=(94,2)]
                                    if((int)fun.isAutorizza()==1){
                    // end
                    out.print(new String(_jspx_html_data[12]));
                    // begin [file=C:/Programmi/IBM/VisualAge for
Java/ide/project_resources/IBM WebSphere Test
Environment/hosts/default_host/default_app/web/delfo/news/home.jsp;from=(1
05,4);to=(107,2)]
                                    }
                    // end
                    out.print(new String(_jspx_html_data[13]));
             } catch (Throwable t) {
                    if ((!_jspx_cleared_due_to_forward) &&
(out.getBufferSize() != 0))
                           out.clear();
                    throw new HandleErrorPageException("errorpage.jsp",
t, out);
             } finally {
                    if (!_jspx_cleared_due_to_forward)
                           out.flush();
                    _jspxFactory.releasePageContext(pageContext);
             }
      }
```

**Figura 50: codice del Servlet generato dalla pagina home.jsp**

La pagina, dopo essere stata tradotta in un Servlet, viene interpretata dall'Application server e dà origine al codice HTML.

Codice HTML generato da home.jsp

```
<html>

<head>
<META HTTP-EQUIV="Expires" CONTENT="Wed, 16 June 1999 00:00:00 GMT">
<META HTTP-EQUIV="pragma" CONTENT="no-cache">
<title>Home page</title>
</head>

 <body bgcolor="#FFFFFF">
   <font face="verdana">
   <font color="navy"><b><p align="center">Ciao delfo, benvenuto nel
motore di News</b></p></font>
   <table align ="center" border ="0" >
   <tr>
   <td bgcolor="#CC9988" colspan="2"><b> </b>
   </td>
   <td>
    <img src="/delfo/News/images/lente.gif">
   </td>
   <td valign="middle">
    <b><a href="/servlet/news.NewsManager?function_news=leggi" alt="leggi
news"><img src="/delfo/News/images/scritta_leggi.gif" border="0"></a>
```

```
    </b><br>
   </td>
   </tr>
   <tr>
   <td>
         <img src="/delfo/News/images/matita.gif">
   </td>
   <td>
   <b>
    <a href="/servlet/news.NewsManager?function_news=crea"><img
src="/delfo/News/images/scritta_nuova_news.gif" border="0">
    </a>
   </b>
    <br>
   </td>
   </tr>
   <tr>
   <td>
         <img src="/delfo/News/images/sbagliate.gif">
   </td>
   <td>
   <b>
   <a href="/servlet/news.NewsManager?function_news=modifica"><img
src="/delfo/News/images/proprie_news.gif" border="0"></a>
   </b>
   <br>
   </td>
   </tr>
   <tr>
   <td>
         <img src="/delfo/News/images/pubblica_grande.gif">
   </td>
   <td>
   <b><a href="/servlet/news.NewsManager?function_news=pubblica"><img
src="/delfo/News/images/scritta_pubblicazione.gif" border="0"></a>
   </b><br>
   </td>
   </tr>
   <tr>
   <td>
         <img src="/delfo/News/images/schedario.gif">
   </td>
   <td>
   <b><a href="/servlet/news.UserManager?function=superuser"><img
src="/delfo/News/images/scritta_autorizzazioni.gif"
border="0"></a></b><br>
  </td>
  </tr>
  <tr>
  <td bgcolor="#CC9988" colspan="2"><b> </b>
  </td>
  <td>
  </td>
  </tr>
  <tr>
  <td colspan="2" align="center"><a
href="/servlet/news.UserManager?function=logout"><img
src="/delfo/News/images/scritta_logout.gif" border="0"></a><br></b>
  </td>
  <td>
```

```
  </td>
   </tr>
   </table>
  </font>
  </font>
 </body>
</html>
```

**Figura 51: codice HTML generato dall'interpretazione di home.jsp**

Come si può notare, il codice è privo dei tag JSP ( <% … %>). Questo poiché il codice racchiuso fra tali tag è già stato interpretato e sostituito dall’HTML prodotto.

## *Menù iniziale personalizzato: home.jsp*

La pagina mostra le opzioni a disposizioni nella pagina iniziale di un utente che dispone di tutti i diritti. Sono disponibili le funzioni:

- ?? lettura
- ?? scrittura
- ?? cancellazione delle news scritte dall'utente
- ?? pubblicazione/cancellazione news di tutti gli autori
- ?? accesso al menù di gestione delle autorizzazione e recupero di news cancellate
- ?? logout dal sistema


**Figura 52: home.jsp visualizzata sul browser**

Naturalmente, utenti in possesso solo di alcune autorizzazioni, interagiscono con un menù privo di alcune voci.

### *Lettura delle news pubblicate: leggi_news.jsp*

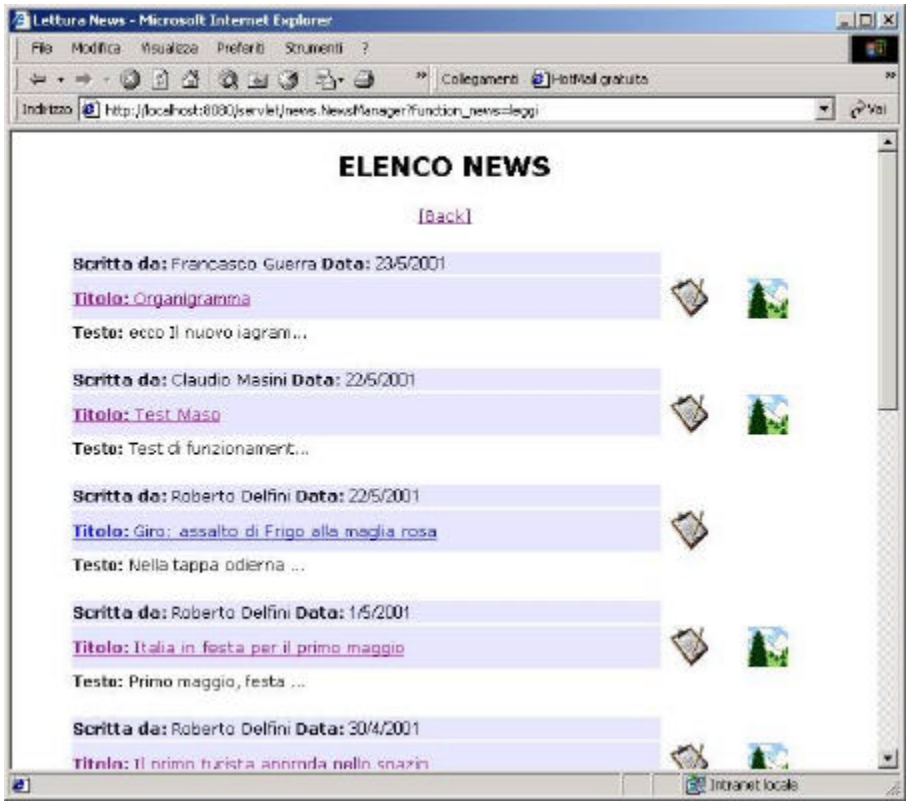


**Figura 53: leggi_news.jsp visualizzata sul browser**

La pagina mostra la lista delle notizie pubblicate, cioè visibili anche agli utenti non registrati. Sulla destra di ogni notizia compaiono delle icone che informano della presenza di link o di un'immagine allegata alla news.

Cliccando sul titolo di ogni notizia si espande la notizia a tutto schermo, visualizzando il testo, l'immagine ed i link.

*Espansione di una news pubblicata: espandi_news.jsp*

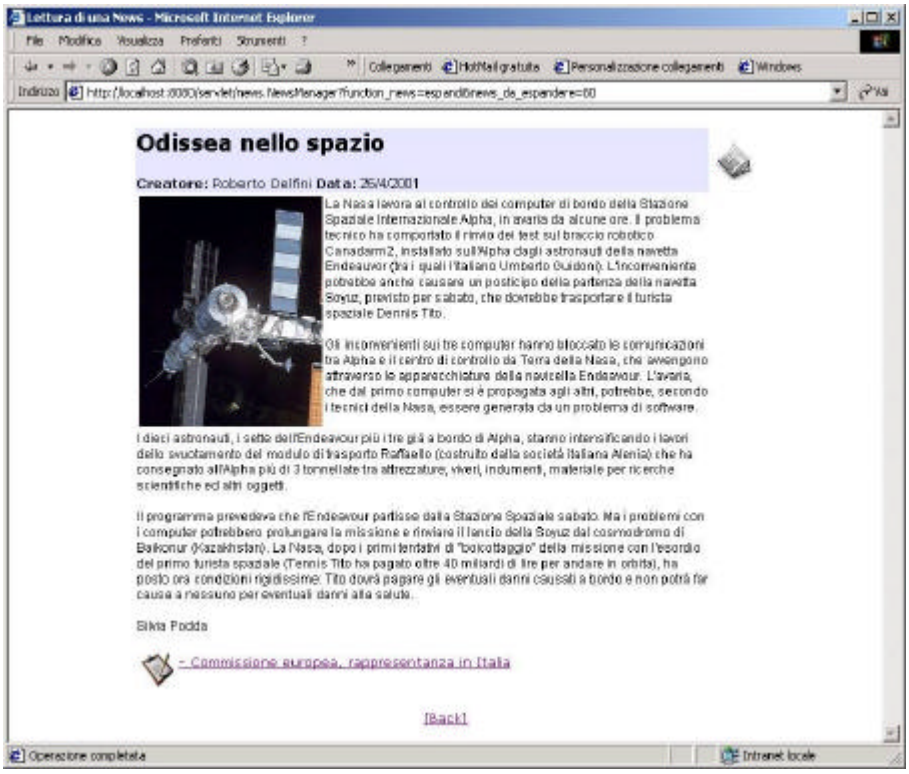


**Figura 54: espandi_news.jsp visualizzata sul browser**

La pagina mostra la notizia espansa a tutto schermo, con tutte le informazioni ad essa allegate.

L'utente non può compiere nessun gesto sulla notizia stessa, se non quello di aprire un collegamento alla pagina lincata.

*Accesso alle funzioni di pubblicazione: pubblica_news.jsp*

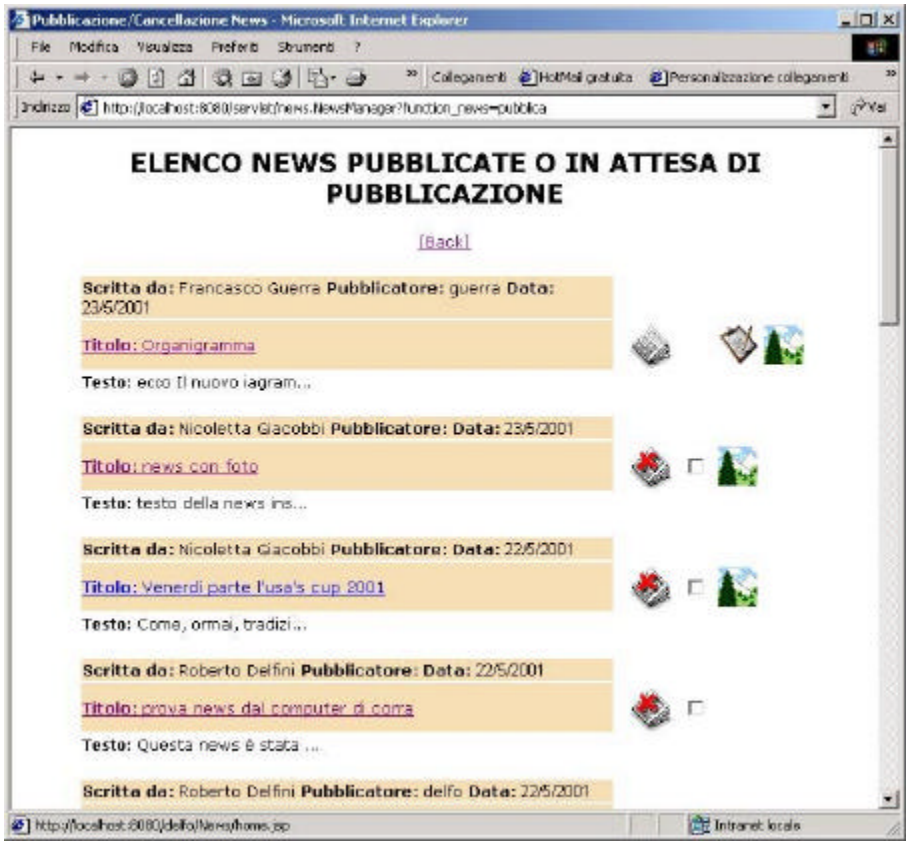


**Figura 55: pubblica_news.jsp visualizzata sul browser**

La pagina mostra l'elenco di tutte le news, pubblicate e non pubblicate affiancando a queste ultime un'icona di un giornale barrato da una croce rossa.

L'utente può scegliere due modi per pubblicare una notizia:

?? Utilizzare la check box posta a destra delle notizie non pubblicate
?? Cliccare sul titolo per ingrandire la notizia e decidere in un secondo tempo la sua pubblicazione o la sua cancellazione.

Rispetto alla pagina pubblica di lettura delle news, in questa pagina viene visualizzata l'informazione dell'username dell'utente che si è occupato della pubblicazione della singola notizia.

### *Espansione di una news da pubblicare: pubblica_espandi.jsp*

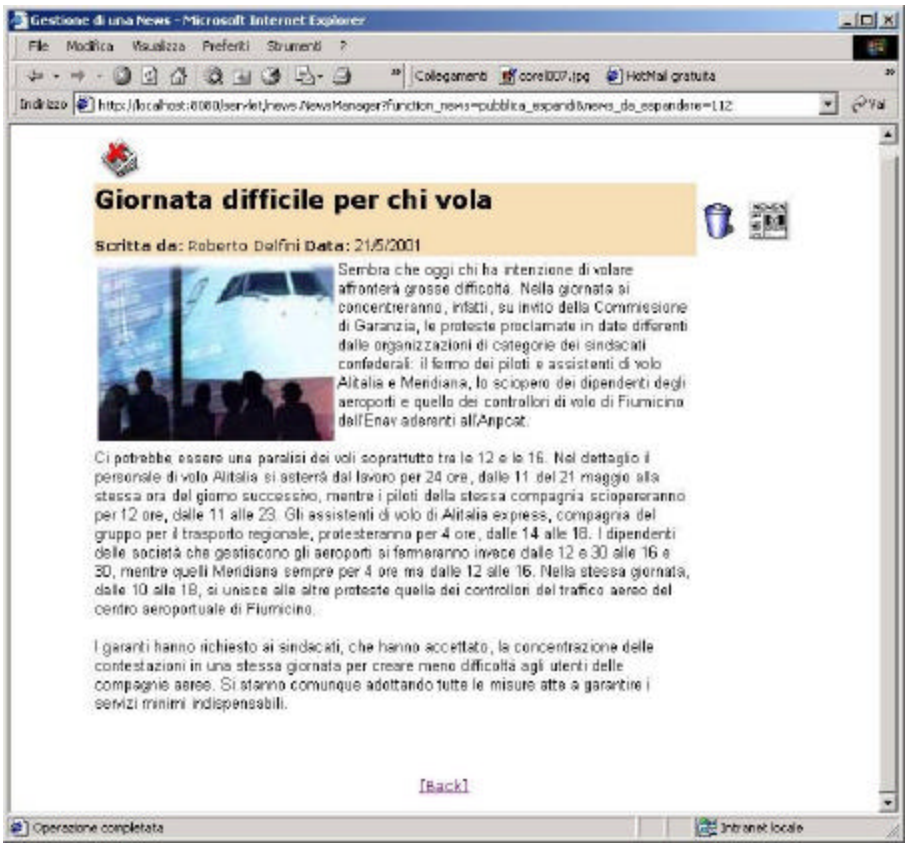


**Figura 56: pubblica_espandi.jsp visualizzata sul browser**

La pagina offre la possibilità di agire sulla notizia, tramite le due icone poste sulla destra del titolo della news.

L'icona del cestino, se cliccata, provoca la cancellazione della notizia, mentre l'icona del giornale effettua la pubblicazione.

Le news cancellate, come già detto, non vengono rimosse dal sistema, ma vengono poste in uno stato particolare, in cui possono essere lette ed

eventualmente recuperate solamente da un SuperUser durante il suo compito di amministrazione del sistema.

### *Cancellazione delle proprie news: modifica_news.jsp*

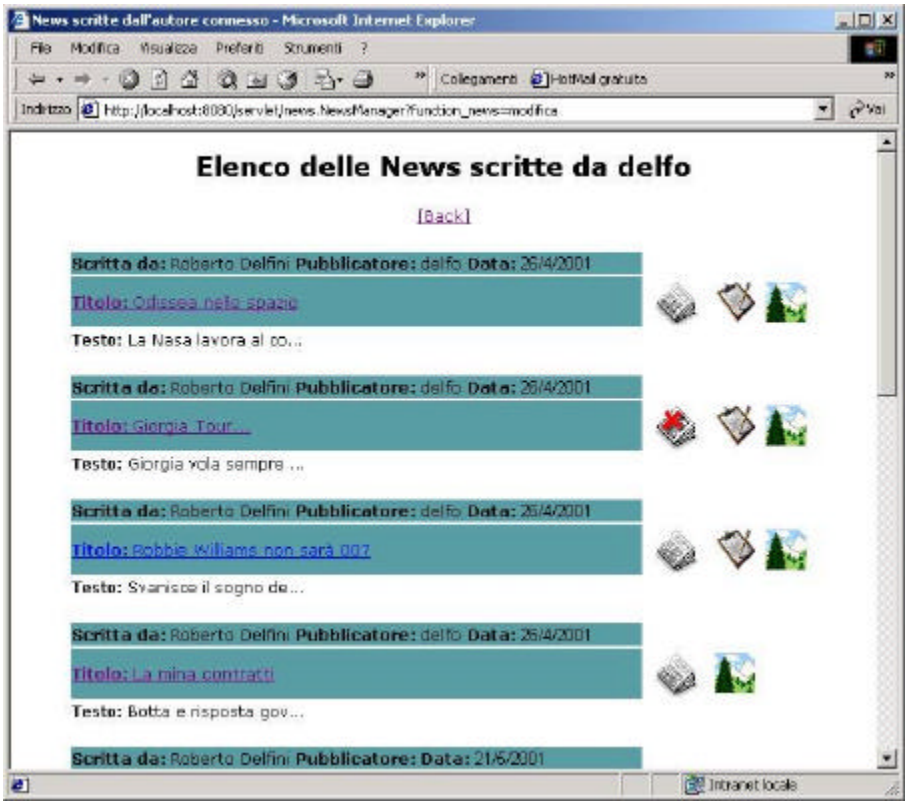


**Figura 57: modifica_news.jsp visualizzata sul browser**

La pagina offre agli utenti che hanno i diritti di creazione, di ottenere una visione d’insieme delle proprie notizie ed eventualmente di cancellarle. Questo si rende necessario in caso di scrittura di informazioni errate da parte del creatore, per evitare di sottoporle alla visione del pubblico.

Per cancellare la news è necessario ingrandirla a tutto schermo ed agire sull’icona posta a destra del titolo.

*Espansione a tutto schermo di una propria news: modifica_espandi.jsp*

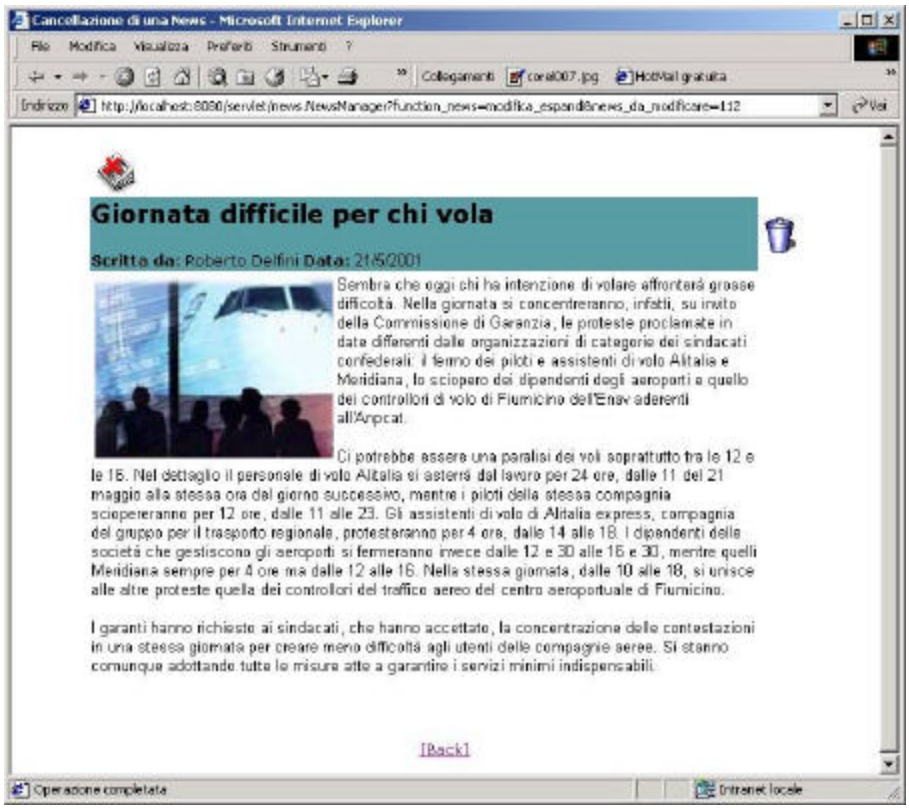


**Figura 58: modifica_espandi.jsp visualizzata sul browser**

La cancellazione della news avviene cliccando sull'icona del cestino posta alla destra del titolo. Nella parte superiore della pagina, un'icona indica lo stato della news (pubblicata/non pubblicata).

*Creazione di una news: crea_news.jsp*

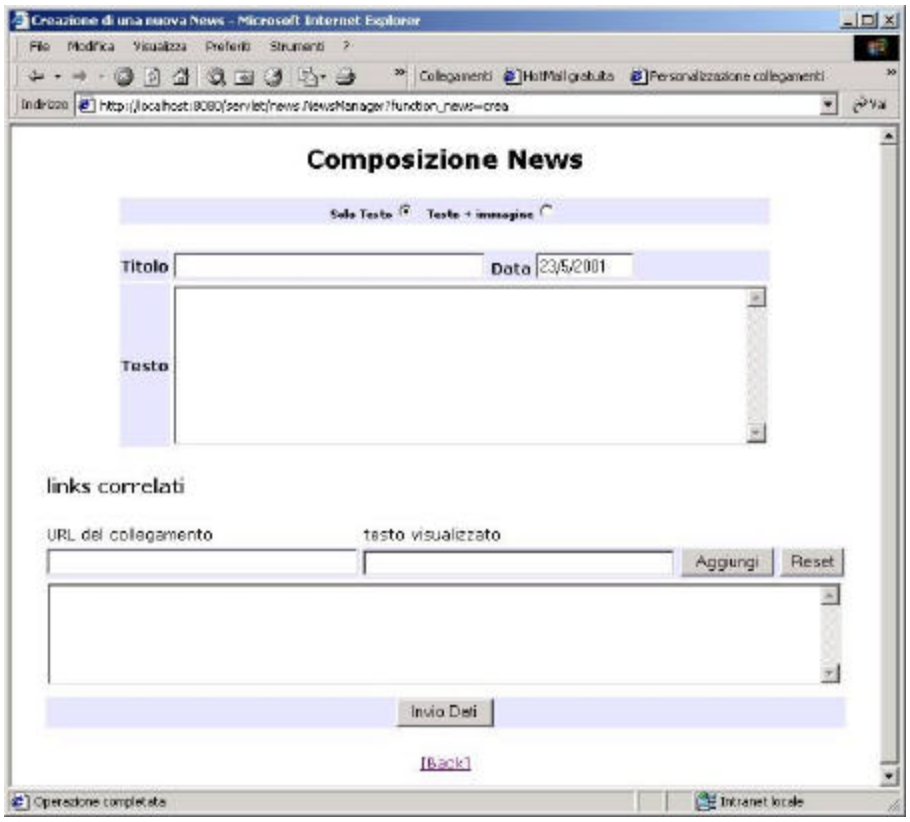


**Figura 59: crea_news.jsp visualizzata sul browser**

Nella parte superiore della pagina è possibile selezionare il formato della notizia che si desidera inserire: solo testo o testo con allegato un'immagine.

Il titolo della notizia e il corpo del testo sono inseribili tramite due text box. Nella parte inferiore della pagina è gestito l'inserimento di link.

Una volta che i dati sono stati inviati, la news è inserita nella base di dati. Nel caso venga selezionato il tipo testo e immagine, l'utente dovrà interagire con una seconda pagina nella quale è possibile sfogliare il file system del proprio computer alla ricerca del file formato JPG da allegare.

Nella pagina sono presenti controlli effettuati con JavaScript che avvertono della necessità di compilare i campi obbligatori *titolo* e *corpo*.

*Menù di amministrazione delle autorizzazioni: superuser.jsp*

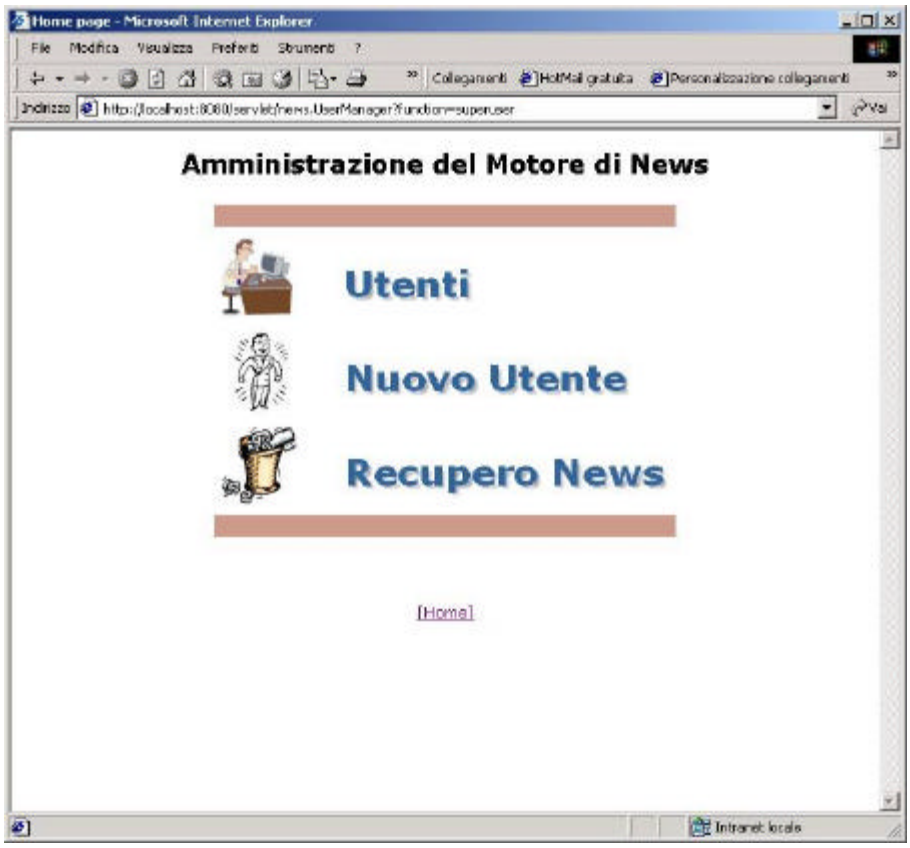


**Figura 60: superuser.jsp visualizzata sul browser**

La pagina in figura è visibile unicamente agli utenti in possesso dei diritti di amministrazione del sistema.

Le funzioni disponibili sono:

?? Visualizzazione di un elenco degli utenti con la possibilità di rimuoverli dal sistema
?? Inserimento di un nuovo utente
?? Recupero di news precedentemente cancellate

*Inserimento di un nuovo utente: nuovo_utente.jsp*

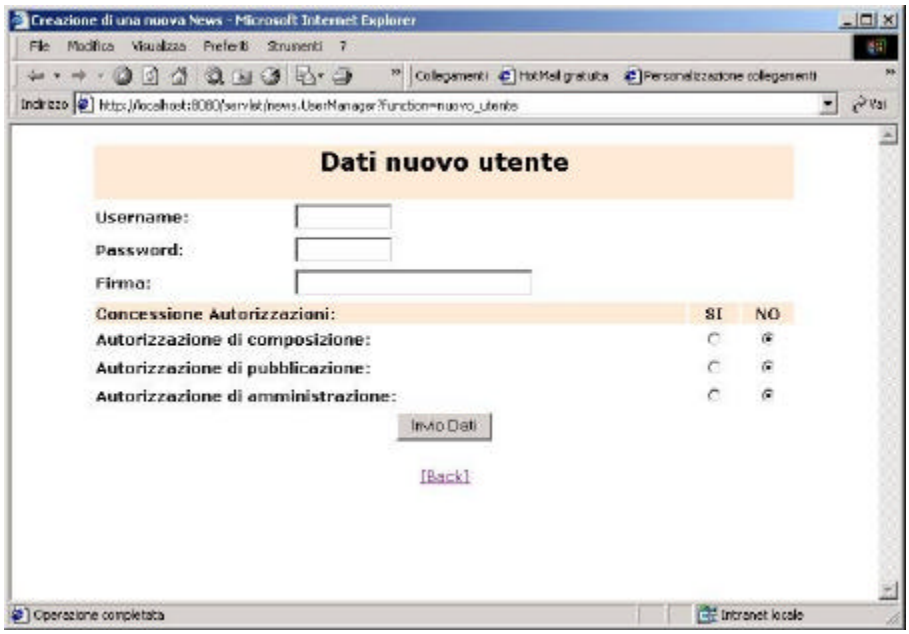


**Figura 61: nuovo_utente.jsp visualizzata sul browser**

Nella parte superiore della pagina vengono immessi i dati dell’utente, nella parte inferiore, tramite check box vengono assegnati i diritti di creazione, pubblicazione e amministrazione. Il diritto di lettura è sottinteso e ne sono in possesso anche gli utenti non registrati.

Una volta che i dati vengono inviati al server, avviene il loro inserimento nella base di dati, e un messaggio informa l’utente del successo dell’operazione.

Ogni utente deve avere uno username univoco all’interno del sistema, quindi nel caso di inserimento di un nuovo utente avente un username già utilizzato, un messaggio provvede ad informare l’amministratore.

A differenza dei controlli effettuati nella pagina di inserimento di una nuova notizia, che girano sul lato client, questo controllo deve accedere al database per verificare l’unicità dello username del nuovo utente. Questo controllo è compito del modulo UserManager e il frammento di codice interessato gira sul lato server.

*Visualizzazione delle news cancellate: recupera_news.jsp*

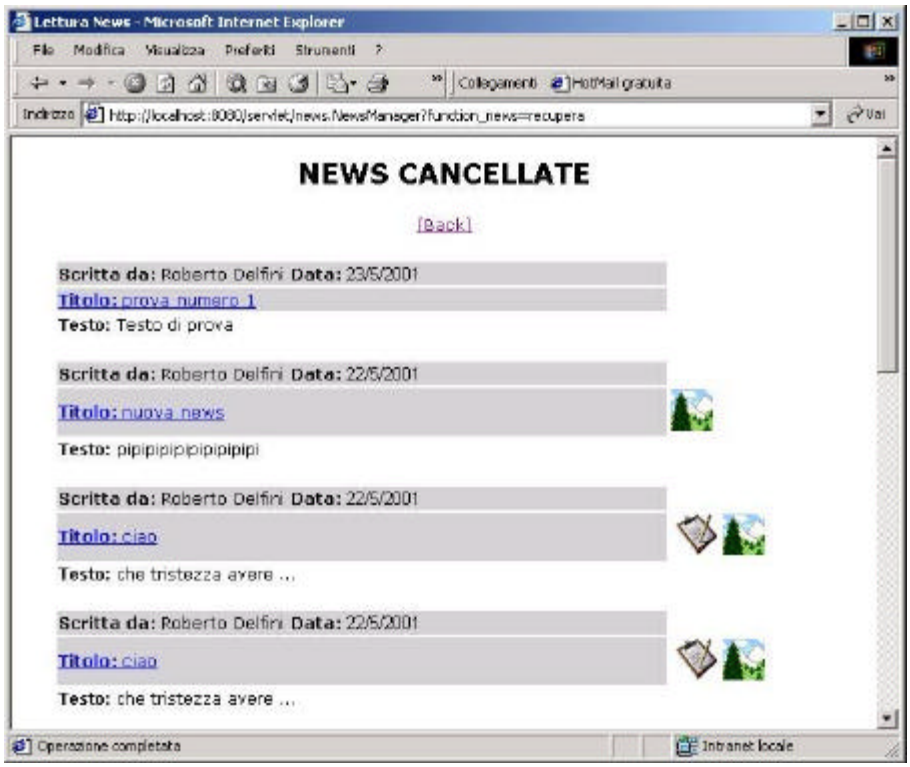


**Figura 62: recupera_news.jsp visualizzata sul browser**

Analogamente a quanto accade per la pubblicazione e per la lettura delle news, a destra del titolo, sono elencate le caratteristiche delle notizie.

Per effettuare l'operazione di *undelete* è necessario espandere la news sotto esame, cliccando sul titolo.

Non è stata implementata nessuna funzione di rimozione della notizia dal sistema, ma la sua realizzazione comporterebbe modifiche irrisorie del modulo NewsManager e sarebbe selezionabile da questa sezione dell'interfaccia utente.

*Espansione di una news cancellata: recupera_espandi.jsp*

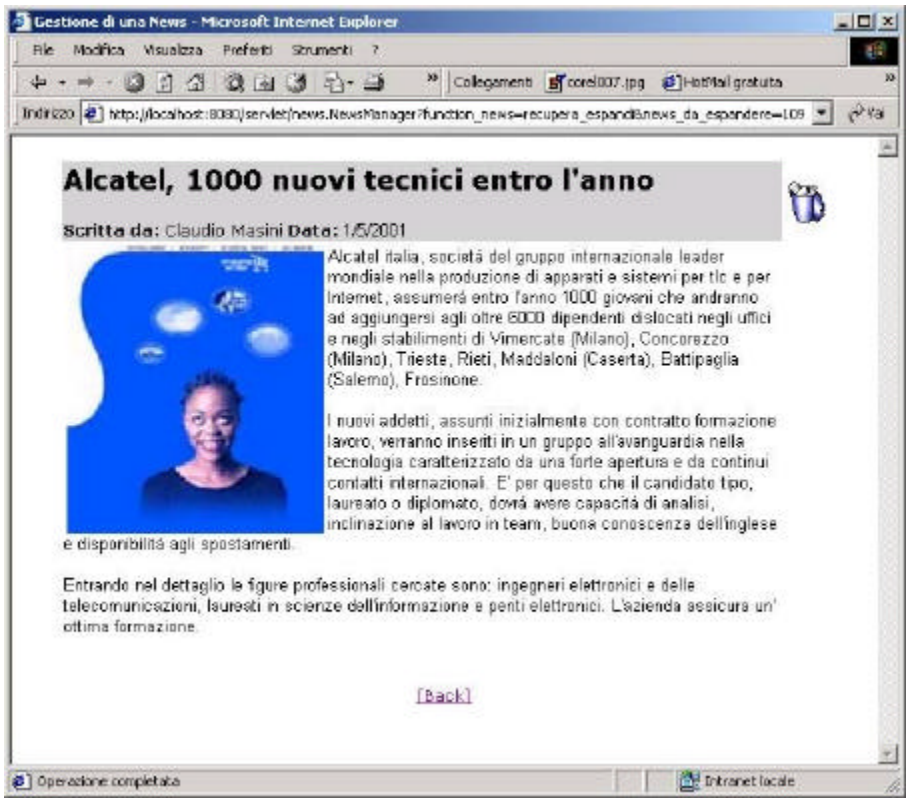


**Figura 63: recupera_espandi.jsp**

La pagina è simile a quelle presentate nelle funzioni di lettura e pubblicazione, ma l'unica azione permessa è quella di recupero della notizia visualizzata, cliccando sull'icona del cestino aperto, sulla destra del titolo.

La notizia, una volta recuperata, viene posta nello stato di *non pubblicata*, quindi non è visibile agli utenti generici, se non dopo avere effettuato l'operazione di pubblicazione.

### 4.2.11 Installazione e test

La realizzazione di un prototipo funzionante ha richiesto una fase di test dell’applicazione per verificare l’effettivo funzionamento di tutte le funzioni implementate.

Per fare ciò si è reso necessario esportare l’applicazione, per trasferirla dall’ambiente di sviluppo ad un ambiente nel quale il motore di news si possa trovare nelle condizioni più simili a quelle di lavoro.

#### *Il database*

La prima preoccupazione è rappresentata dal database. L’intero motore di news è stato sviluppato utilizzando Microsoft Access come database. Le potenzialità dell’applicazione e la sua natura multiutente, hanno reso necessaria l’esportazione delle tabelle da Microsoft Access a IBM DB2.

Questa operazione ha permesso di verificare l’effettiva portabilità della classe Dbmanager, richiedendo unicamente la modifica del valore di quattro variabili di classe.:

- driver
- connessione
- db_username
- db_password

In figura sono evidenziate le due variabili da modificare, il passaggio da un database all’altro si effettua sostituendo le stringhe per l’interfacciamento ad Access con le opportune stringhe che permettono l’interfacciamento a DB2.

```
public class DBmanager {

       static String driver="sun.jdbc.odbc.JdbcOdbcDriver";

       static String connessione="jdbc:odbc:DataBaseNews";

       static String db_username = "";

       static String db_password = "";

. . . . . .
. . . . . .
```

**Figura 64: variabili responsabili della connessione verso il database**

Anche se la riscrittura dei valori attribuiti a queste quattro variabili, non comporta grandi sforzi, sarebbe auspicabile, in una versione più evoluta dell’applicazione, di effettuare la configurazione automatica di tali valori.

Naturalmente, visto che tali variabili permettono la connessione con un database, è impossibile memorizzare tali dati in una tabella di quest’ultimo. La soluzione è fare in modo che l’applicazione vada a leggere tali valori in un file di testo al momento dell’istallazione.

L’utente istallatore avrà il compito di modificare questo file di configurazione come già in uso in moltissime applicazioni commerciali.

Questo permetterebbe di non dover più mettere mano ai file sorgente delle classi, evitandone la ricompilazione.

## *L’application server*

Lapplication server utilizzato durante lo sviluppo del software fa parte di un più complesso ambiente di sviluppo descritto in appendice.

Per massimizzare le prestazioni del software, le classi prodotte sono state compilate ed esportate in file bytecode per poi essere trasferiti nelle cartelle di Tomcat predisposte ad accogliere i servlet e le classi.

Operazione analoga è stata effettuata per le pagine JSP. L’unica accortezza richiesta nell’esportazione delle pagine consiste nel riprodurre la struttura di direttori utilizzata durante lo sviluppo, in modo che sia mantenuta la validità dei percorsi relativi utilizzati per l’apertura delle pagine.

L’applicazione, all’atto dell’inserimento di news con immagini, provvederà automaticamente alla creazione di una cartella personale per

ogni utente, in cui vengono conservati i files relativi alle immagini allegate alle notizie.

Questa soluzione comporta la presenza di una struttura di cartelle tanto ampia, quanti sono gli utenti che hanno inserito notizie con immagine.

Ogni immagine avrà un nome che corrisponde a quello che aveva nel client, preceduto dall’identificatore numerico della news a cui appartiene.

Anche nelle pagine Jsp vengono utilizzati riferimenti tramite percorsi relativi ai file contenuti in queste directory, quindi la cancellazione di uno di questi file corrisponde alla privazione di una news della sua immagine.

La scelta della memorizzazione delle immagini nel file system è stata dettata dall’esigenza di massimizzare la portabilità del sistema attraverso differenti DBMS.

## *Le prestazioni*

Al termine delle operazioni d’installazione, è stato verificato un rilevante aumento della velocità dell’applicazione rispetto all’ambiente di test ed un corretto svolgimento delle funzioni richieste.

Sono state effettuate prove di accesso concorrente e interventi di modifica simultanea degli stessi dati. In tutti i casi il sistema funziona correttamente rispettando la consistenza e l’integrità dei dati stessi.

Questo permette di utilizzare il prototipo realizzato per la pubblicazione di news in siti *monodominio* in cui il numero di accessi dipende fortemente dal DBMS utilizzato e dalla potenza di calcolo del server.

## *Ampliamento delle funzionalità*

Le funzioni offerte dal software realizzato sono le minime indispensabili per un motore di news. La scelta di implementare solo questo tipo di funzionalità è stata dettata dalla volontà di concentrare gli sforzi sull’aspetto progettuale tralasciando quei particolari accessori che rendono più gradevole l’uso all’utente.

La struttura modulare con cui è stato progettato il motore di news, lascia ampio spazio all’aggiunta di funzioni accessorie. Per alcune di esse è già stato scritto parte codice in modo che possano essere ospitate in una futura versione.

Le funzioni gia previste sono:

- ?? Possibilità di scegliere non più fra due stili di composizione di notizie, ma fornire la libertà di aggiungere una, due o più foto per ogni news.
- ?? Possibilità di scegliere fra differenti template grafici di visualizzazione della stessa notizia, in modo da fornire una sorta di personalizzazione sul formato della pagina.
- ?? Scelta fra differenti ordinamenti delle news in fase di lettura.
- ?? Funzione di cancellazione fisica di una notizia dal sistema, senza la possibilità di recuperarla in un secondo tempo.
- ?? Funzione di autocancellazione delle notizie pubblicate dopo un certo numero di giorni dalla data di creazione.

Altre caratteristiche non sono state previste, ma richiederebbero modeste modifiche al codice, localizzate all’interno dei moduli di competenza.

Le funzioni facilmente implementabili sono:

- ?? Differente interfaccia utente che permetta, a chi è in possesso dei diritti necessari, di pubblicare una notizia direttamente dopo la fase di composizione.
- ?? Presentazione delle notizie suddivise per pagine all’atto della lettura.
- ?? Aggiunta di attributi ad ogni notizia con più informazioni in fase di visualizzazione.
- ?? Creazione di una struttura che permetta di memorizzare i dati anagrafici di ogni utente creatore ed eventualmente allegare alcuni di essi alla visualizzazione delle news secondo particolari esigenze.

# Conclusioni

Durante il lavoro svolto sono state esaminate le problematiche relative al progetto ed all'implementazione di un'applicazione server-side per la pubblicazione automatica di notizie sul Web.

L'analisi del problema, per come è stata compiuta, ha permesso di attraversare in successione tutte le fasi caratteristiche della nascita di un software ed ha permesso di scontrarsi con l'esigenza di effettuare scelte vincolanti per tutta la durata del progetto e di valutare in prima persona la validità di tali decisioni.

Capitolo dopo capitolo sono state presentate tutte le motivazioni che hanno portato a preferire determinate soluzioni rispetto ad altre, dalla tecnologia utilizzata, al linguaggio di modellazione, fino ad arrivare alle specifiche soluzioni implementative.

La modellazione in UML ha portato ad una chiara organizzazione dei moduli dell'applicazione e ad una più facile traduzione in codice delle entità descritte.

L'utilizzo di Java come base tecnologica su cui sviluppare l'applicazione ha confermato la potenza e la versatilità di tale tecnologia, soprattutto per quanto concerne lo sviluppo di applicazioni orientate al Web e alla generazione di pagine dinamiche.

In conclusione possiamo affermare che la scelta di occuparsi degli aspetti legati alla progettazione e la realizzazione del motore di news, si è rivelata un'occasione per affrontare un problema 'moderno'.

La tecnologia utilizzata, il reperimento dei dati da un database, l'alto contenuto dinamico delle pagine JSP, lo studio della portabilità e della personalizzazione dell'applicazione, sono elementi che caratterizzano un software dell'ultima generazione.

Il vero aspetto che attribuisce modernità al lavoro svolto è la filosofia che è all'origine del motore di news, cioè l'intento di rendere la tecnologia utilizzabile da un numero sempre più elevato di persone e di conseguenza fornire un'auspicabile indipendenza a coloro che si servono di un prodotto dedicato alla rete.

Questo porta a nobilitare le mansioni degli operatori del Web, troppo spesso impegnati a produrre inutile HTML statico o a compiere sterili manutenzioni di prodotti ormai obsoleti.

Inoltre il motore di news si è dimostrato un buon pretesto per affrontare uno studio approfondito di nuove tecnologie destinate ad interpretare il ruolo di protagonista nello scenario informatico di un imminente futuro.

# Appendice A

# L’ambiente di sviluppo

## A.1 Il Java Developer’s Kit

Java è stato presentato al mercato come l’uovo di Colombo per i problemi della portabilità del software. Per questo motivo, la Sun ha sviluppato un pacchetto dedicato ai programmatori che si mantiene pressoché uguale nelle versioni per tutte le piattaforme.

Il minimo necessario per sviluppare applicazioni in Java è composto dal JDK (Java Developer’s Kit). Il suo scopo è quello di fornire un insieme completo di strumenti per lo sviluppo, la documentazione e l’esecuzione di programmi e applet Java. Il JDK fornisce strumenti che supportano ciascuna di queste attività ed altre ancora.

È costituito da sette programmi:

| **Nome Programma** | **Descrizione** |
|---|---|
| javac | Compilatore |
| java | Interprete |
| jdb | Debugger |
| javap | Disassembler |
| appletviewer | Visualizzatore di applet |
| javadoc | Generatore di documentazione |
| javah | Generatore di file header C |

Normalmente si scrivono i programmi Java utilizzando un editor di testo per sviluppare files sorgente Java. Questi files consistono di pacchetti di codice sorgente che dichiarano classi e interfacce Java. I file sorgente hanno estensione *.java*. Il compilatore Java (*javac*) viene utilizzato per convertire i files sorgente in files che possono essere eseguiti utilizzando

l’interprete di Java. Questi files sono detti *bytecode files* e terminano con l’estensione *.class*.

L’interprete Java esegue classi contenute nei file bytecode. Esso verifica l’integrità, la correttezza operativa e la sicurezza di ciascuna classe nel momento in cui questa viene caricata ed eseguita interagendo con il sistema operativo, con l’ambiente di gestione delle finestre e con le utilità di comunicazione per produrre il comportamento desiderato del programma.

Il debugger (*jdb*) è analogo all’interprete in quanto esegue classi Java che sono state compilate in file bytecode, ma fornisce anche speciali capacità di interrompere l’esecuzione del programma in punto selezionati e visualizzare il valore delle variabili delle classi. Queste capacità sono indispensabili per individuare e correggere gli errori concettuali effettuati durante la programmazione.

Il disassembler prende i file bytecode e visualizza le classi, le variabili e i metodi che sono stati compilati. Esso identifica inoltre le istruzioni bytecode utilizzate per implementare ciascun metodo.

Il visualizzatore di applet (*appletviewer*) visualizza le applet java contenute in pagine Web, poste nel proprio sistema di file locali o in siti accessibili. Viene inoltre utilizzato per testare le proprie applet sviluppate.

Lo strumento di documentazione automatica (*javadoc*) è utilizzato per convertire parti dei files sorgente di Java in HTML. I files HTML generati documentano le classi, le variabili e i metodi, le interfacce e le eccezioni contenute nel file sorgente, basandosi su commenti scritti in un particolare formato.

Lo strumento file di intestazione C (*javah*) è utilizzato per generare intestazioni e file sorgente in linguaggio C da un file bytecode Java. I file generati da javah sono utilizzati per sviluppare metodi nativi, cioè classi Java scritte in un linguaggio diverso da Java.

Ogni programma viene invocato da riga di comando e il JDK non fornisce nessun editor di testo per produrre codice.

## A.2 IBM Visual Age for Java

Nonostante la completezza del JDK, spesso si avverte l’esigenza di un maggiore supporto durante la fase di sviluppo e di test del software.

Il primo limite sorge nell’editor utilizzato per scrivere i files sorgente. Spesso i moderni editor offrono la possibilità di aumentare la leggibilità

del codice con opportune indentazioni e con diversi colori ai frammenti di codice che svolgono differenti funzionalità, rispetto i semplici editor di testo come il ‘blocco note’ di Windows o il ‘VI editor’ di Unix.

Così sono nati veri e propri ambienti di sviluppo, completi di strumenti per rendere più veloce e confortevole la stesura del codice.

L’ambiente di sviluppo utilizzato per la produzione del codice del motore di news è *IBM Visual Age for Java*.

Questo strumento mette lo sviluppatore nelle migliori condizioni per produrre codice offrendo le seguenti possibilità a completamento di quelle offerte dal JDK:

- ?? Ambiente predisposto per lo sviluppo collettivo di applicazioni, con la possibilità di lavorare in più programmatori sulla stessa applicazione con la gestione degli aggiornamenti del software in costruzione e l’integrazione automatica del lavoro svolto
- ?? Efficienti strumenti per il riutilizzo di codice, con la gestione di un *repository* in cui vengono conservate le classi prodotte, ordinate per progetto e package.
- ?? Per gli sviluppatori più inesperti offre uno scenario per *e-business* per mezzo del quale lo sviluppatore impara come creare nuove applicazioni semplicemente modificando e personalizzando soluzioni precostruite. Grazie al *tool integrator API* lo sviluppo è completamente assistito con funzioni di completamento automatico durante la digitazione e visualizzazione istantanea dei metodi disponibili per la classe in uso.
- ?? Generazione automatica del codice dei metodi accessori relativi agli attributi delle classi e gestione assistita delle eccezioni.
- ?? Compilazione automatica con segnalazione degli errori e visualizzazione della parte di codice in cui è presente il problema.
- ?? Debug avanzato ed assistito con funzioni di esplorazione dei diversi livelli in cui viene interpretato il codice, con possibilità di addentrarsi all’interno dei blocchi e dei metodi durante il debug.
- ?? Utilizzo di breakpoint multipli e visione dettagliata delle variabili e del loro valore nel corso dell’esecuzione.
- ?? Test delle applicazione Web assistito e facilitato grazie all’integrazione dell’application server *Web Sphere Test Environment* che include, fra le tante funzioni, la possibilità di effettuare il debug delle pagine JSP e le altre tecnologie *server-side* come se fossero normali classi Java.

Tutto ciò consente di superare i limiti del JDK e di migliorare la qualità ed il tempo di sviluppo a vantaggio del costo finale dell’applicazione.

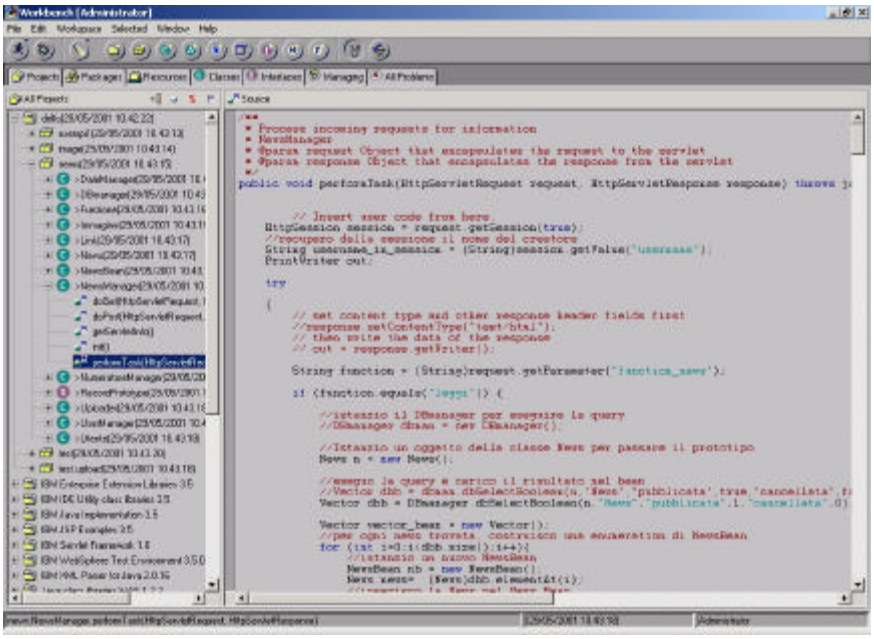

**Figura 65: spazio di lavoro (workbench) di Visual Age for Java. Sulla sinistra sono visibili i package e le classi, mentre sulla destra è visualizzato il codice del metodo selezionato.**

La presenza di *Web Sphere Test Environment Application Server* ha permesso di effettuare fin dalle prime fasi dello sviluppo del motore di news, un test accurato dell’applicazione senza preoccuparsi dell’esportazione delle classi in altri application server.

L’utilizzo di numerose pagine JSP ha reso necessario l’utilizzo della funzione di debug sulle pagine stesse.

Naturalmente le prestazioni di *Web Sphere Test Environment* non sono confrontabili in termini di velocità e sicurezza con quelle di un Application server come Tomcat. Questo è dovuto alle finalità diverse dei due prodotti.

Lo scopo del *Test Environment* è quello di permettere una prima visione del risultato del lavoro effettuato e di fornire un ambiente grafico assistito in cui lavorare più facilmente e velocemente durante la scrittura del codice.

Per il funzionamento dell’applicazione è indispensabile uscire dall’ambiente di test ed esportare le classi e le pagine JSP sotto il controllo di un ‘vero’ application server al fine di massimizzare le prestazioni e rendere il software effettivamente utilizzabile.

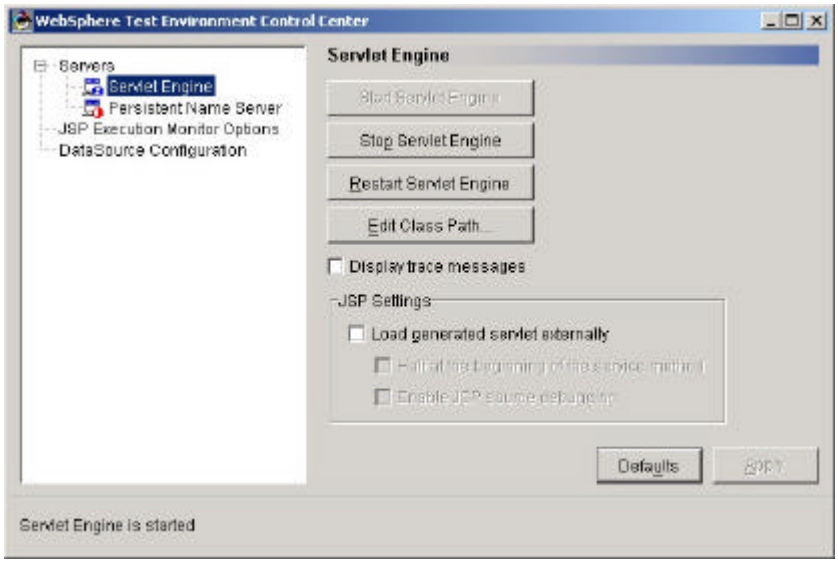


**Figura 66: pannello di controllo di WebSphere Application Server**

Durante lo sviluppo dell’applicazione del motore di news si è reso necessario passare da piattaforma Unix a sistema operativo Windows. Questo ha dettato la necessità di utilizzare il JDK in ambiente Unix e Visual Age in ambiente Windows. La differenza per quanto concerne la velocità ed il comfort di sviluppo si è rivelata notevole.

È indispensabile che ogni programmatore di qualsiasi livello, venga messo nelle condizioni migliori per operare, valutando di volta in volta la disponibilità di strumenti sempre più potenti.

L’ambiente di sviluppo rappresenta, nel caso di uno sviluppatore inesperto, una sorta di tutore che propone esempi e soluzioni alternative, mentre per un utilizzatore più esperto, si trasforma in un prezioso strumento per risparmiare tempo e fatica al fine di concentrare gli sforzi su problematiche di più alto livello.

In entrambe le situazioni descritte, la qualità del prodotto sviluppato migliora notevolmente.

# Appendice B

# Manuale utente

Benché l'interfaccia utente dell'applicazione risulti piuttosto intuitiva, è doveroso spendere alcune pagine per descrivere il corretto modo di utilizzo dell'applicazione.

## B.1 Login

La pagina che permette di effettuare il login degli utenti registrati, è interamente statica, in modo da poter essere accorpata a qualunque pagina HTML che abbia spazio per ospitare i due piccoli campi di testo e il pulsante di invio dei dati.

Dopo la fase di login, compare un menù personalizzato, la cui versione completa è presentata in cap. 4 fig. 52.

L'utente a questo punto può selezionare la funzione desiderata cliccando sulla voce corrispondente.

## B.2 Lettura News

Selezionando la funzione di lettura delle news, si apre una pagina simile a quella mostrata in cap 4 fig 53.

La pagina mostra le news già pubblicate, cioè disponibili alla visione di utenti registrati e non registrati, infatti questa pagina è l'unica, fra quelle visualizzate dinamicamente dal motore di news, che può essere vista anche senza effettuare il login.

Tipicamente, la funzione di lettura delle news può essere raggiunta dalla home page del sito che utilizza il motore o addirittura accorpata in essa.

Per ogni notizia vengono visualizzate due barre color lavanda. La prima indica il nome del creatore e la data della scrittura della notizia, la seconda mostra il titolo della news sul quale è possibile cliccare per ingrandire la notizia a tutto schermo per una facile lettura.

Sulla destra di tali barre possono essere presenti alcuni simboli che forniscono informazioni sul contenuto della news, precisamente:

**Figura 67: icona che indica la presenza di link**

**Figura 68: icona che indica la presenza di un'immagine allegata**

Sotto il titolo dell'intera pagina e sul fondo di essa è presente il link alla pagina precedente nella quale è visualizzato il menù delle funzioni.

Sotto le barre color lavanda, si può leggere un estratto del testo di ogni notizia.

Cliccando sul titolo di ogni news è possibile espandere la singola notizia a tutto schermo. Con tale azione si apre una pagina simile a quella presentata in cap.4 fig 54.

Nella parte alta della pagina di visualizzazione della singola notizia a tutto schermo, compare una barra color lavanda in cui è mostrato il titolo della news. Sotto tale barra viene mostrato il testo della notizia ed eventualmente un'immagine ad essa allegata, visualizzata in alto a sinistra.

Sotto il testo sono mostrati gli eventuali links correlati alla news, presentati per mezzo di un'icona ed una riga di commento. Cliccando sulla riga di commento viene aperta la pagina collegata in una nuova finestra del browser utilizzato.

Il tasto [back] posto in fondo alla pagina permette di ritornare alla pagina contenente l'elenco di tutte le news.

## B.3 Nuova News

Selezionando dal menù iniziale la funzione di creazione di una nuova news, si apre una pagina uguale a quella mostrata in cap. 4 fig 59.

Analizzando la pagina dall'alto in basso troviamo, subito sotto l'intestazione, due radio-button che permettono la scelta del formato della news che si andrà a scrivere, ovvero: solo testo oppure testo e immagine.

In seguito viene richiesto il titolo della news (obbligatorio) e la data di creazione (campo già precompilato con il valore della data di sistema fornita dal server).

La text-box al centro della pagina ospita il testo della news.

Nella parte bassa della pagina vengono inseriti gli eventuali link correlati alla news.

L'inserimento deve avvenire in fasi distinte:

1. Compilare del campo relativo all'URL del collegamento, ad esempio: www.unimo.it. È necessario evitare di inserire spazi all'interno dell'URL e non inserire caratteri speciali come: *, @, #.
2. Compilare il campo di commento che rappresenta il nome del link che verrà visualizzato nella notizia. Es: - Il sito dell'università. Anche in questo caso è necessario non introdurre il carattere '*' poiché utilizzato dall'applicazione per separare i diversi link fra loro.
3. Cliccare sul tasto 'aggiungi' o sul tasto 'reset' a seconda che si voglia aggiungere il link alla lista o cancellare la lista.

Nella text-box sul fondo della pagina è visionabile la lista dei link con i relativi commenti.

Dopo l'invio dei dati la news viene inserita nel database e, in caso di immagine allegata viene data la possibilità di sfogliare le cartelle del proprio computer per cercare l'immagine da allegare.

In questo caso l'immagine allegata deve obbligatoriamente essere un file JPEG[10] a causa della routine di ridimensionamento che opera solo su file di questo formato.

Dopo i messaggi di conferma dell'operazione avvenuta, si ritorna al menù principale.

[10] Formato di file contenenti immagini compresse con un particolare algoritmo. Tali file hanno l'estensione .jpg.

## B.4 Proprie News

Questa funzione permette di tenere sotto controllo le news inserite dall'utente collegato, sia quelle pubblicate che quelle non ancora pubblicate. La funzione è stata implementata per dare la possibilità di cancellare le news che, in un secondo tempo, sono state ritenute errate dal creatore onde evitare la visualizzazione in rete di dati non corretti.

Selezionando la funzione 'Proprie News' viene visualizzata una pagina simile a quella mostrata in cap.4 fig.57.

La pagina è sostanzialmente dello stesso tipo di quella mostrata per la lettura delle news, ad eccezione del colore delle barre che è verde inglese invece che color lavanda.

Inoltre, fra le icone che visualizzano informazioni sulle singole notizie, compare l'indicazione dello stato della news: pubblicata o non pubblicata.

Anche in questo caso cliccando sul titolo delle notizie, si apre a tutto schermo la notizia.

La pagina è simile a quella mostrata in cap 4 fig.58. Cliccando sull'icona a forma di cestino sulla destra della barra verde del titolo, è possibile cancellare la news, o meglio, porla nello stato di cancellata.

**Figura 69: icona per la cancellazione delle news**

La notizia potrà essere recuperata da un utente in possesso di diritti di superuser (amministratore).

## B.5 Pubblicazione News

Selezionando la funzione di pubblicazione si apre una pagina che mostra l'elenco di tutte le news, pubblicate e non pubblicate scritte da tutti gli utenti creatori.

La pagina mostra, per ogni news, due barre di colore giallo, nella prima delle quali è mostrato il nome del creatore, la data di creazione e lo

username di colui che ha pubblicato la news. Nella seconda barra viene indicato il titolo della news.

A destra di tali barre viene indicato lo stato delle news tramite le solite icone, la presenza di link, di immagini e una check-box.

Agendo su tale check-box è possibile pubblicare in un solo gesto tutte le news contrassegnate, semplicemente agendo sul pulsante ‘pubblica’ in fondo alla pagina.

Anche in questo caso, cliccando sul titolo è possibile espandere la news a tutto schermo per leggerne il contenuto e cancellarla o pubblicarla agendo sulle icone del cestino o del giornale aperto poste sulla destra del titolo.

**Figura 70: icona per cancellazione delle news**


**Figura 71: icona per la pubblicazione delle news**

**Figura 72: icona che indica lo stato di 'pubblicata' della news**

**Figura 73: icona che indica lo stato di 'non pubblicata' della news**

## B.6 Autorizzazioni

Cliccando la voce ‘Autorizzazioni’ compare il menù delle funzioni attribuite al superuser, ovvero l’amministratore del sistema.

Il menù offre le seguenti funzioni:

?? Utenti
?? Nuovo Utente
?? Recupero News

## Utenti

Selezionando questa funzione viene visualizzata una pagina in cui sono mostrati i dati relativi a ciascun utente. Cliccando sull'icona del cestino a fianco ad ogni utente è possibile rimuoverlo totalmente dal sistema, senza che l'azione abbia effetto sulle news precedentemente inserite.

## Nuovo Utente

La pagina che compare contiene la maschera da compilare per introdurre un nuovo utente nel sistema.

Vengono richiesti i dati relativi allo *username*, la *password* e la *firma* che viene visualizzata sulle news.

Tramite dei radio-button vengono attribuiti i diritti di *scrittura*, *pubblicazione* di notizie e i diritti di *superutente*.

## Recupero News

Selezionando questa funzione compare una pagina che mostra l'elenco di tutte le news che sono state cancellate.

Il colore utilizzato per le barre contenenti creatore, data di creazione e titolo di ogni news è il grigio.

Cliccando sul titolo si apre una pagina in cui la news viene mostrata a tutto schermo dove, cliccando sull'icona posta sulla destra del titolo è possibile recuperare la notizia e porla nello stato di *non pubblicata*.

**Figura 74: icona per il recupero delle news**

Dopo ogni operazione di pubblicazione, di cancellazione, di inserimento e di recupero di notizie, vengono visualizzati messaggi di conferma.

## B.7 Logout

Dal menù principale si accede alla funzione di logout che permette di uscire in modo corretto dal sistema.

È importante effettuare il logout per fare in modo che l'applicazione chiuda nel modo corretto la sessione utente.

Non effettuando il logout si corre il pericolo che un utente che richiami le stesse pagine dallo stesso computer entro il tempo di timeout della sessione (30 minuti), possa accedere liberamente al sistema con l'identità di che ha precedentemente utilizzato il motore di news.

Questo è un problema di tutte le applicazioni che sfruttano l'oggetto sessione per lo scambio di dati.

# Bibliografia

JAVA™ 2 Platform, Enterprise Edition Blueprints
*Design Patterns Model-View-Controller (MVC)*
http://www.java.sun.com/j2ee/blueprints

IBM Software
*VisualAge For Java, overview*
http://www-3.ibm.com/software/ad/vajava

Grady Booch, Ivar Jacobson, James Rumbaugh
*The Unified Software Development Process*
Addison Wesley

Martin Fower, Kendall Scott
*UML distilled, quick reference to Object Modelling Language Standard*
Addison-Wesley

James Goodwill
*JSP Java Server Pages, guida di riferimento*
Apogeo

David Flanagan
*JavaScript*
O'Reilly-Apogeo

Giorgio Braga
*Magic JavaScript*
Mondadori informatica