Facoltà di Ingegneria di Como
Politecnico di Milano

Annotazioni di documenti multimediali. Aspetti di cooperazione e personalizzazione.

Relatore:

Prof. Dr. Stefano Ceri
Politecnico di Milano
Dipartimento di Elettronica e Informazione

Correlatore:

Prof. Dr. Joachim W. Schmidt
Arbeitsbereich Softwaresysteme
TU Hamburg-Harburg Deutschland

Tesi di laurea di:

Amanzio Rigamonti
Via Montello n.11
23895 Nibionno, Lecco
Matr. 622705
Facoltà di Ingegneria di Como
Politecnico di Milano

11 febbraio 1998
Technische Universität Hamburg-Harburg

Ringraziamenti

Sono molte le persone che vorrei ricordare e che mi sono state vicine nel periodo che ho trascorso ad Hamburg durante lo svolgimento del mio lavoro.

Un ringraziamento va a tutte le persone del Dipartimento di Informatica della Technische Universität Hamburg-Harburg, che hanno contribuito alla realizzazione del mio lavoro e che mi hanno aiutato pazientemente a superare le numerose difficoltà legate alla diversità di lingua.

In particolare ringrazio il prof. Stefano Ceri ed il prof. Joachim Schmidt, la cui conoscenza ed amicizia mi hanno permesso di fare questa esperienza indimenticabile. Un ricordo va inoltre a Claudia Niederée ed Hans-Werner Sehring, i cui consigli e suggerimenti sono stati per me di estrema utilità ed importanza.

Per ultimo, e proprio per questo il più importante, un ringraziamento di cuore va a tre persone per me speciali; due sono i miei genitori, che ringrazio per l'educazione che mi hanno dato senza la quale nulla di tutto questo sarebbe stato possibile. A loro, alle rinucie ed ai sacrifici che per anni hanno fatto per me va il mio più caro e sincero riconoscimento.

Infine ringrazio infinitamente Anna, per essermi stata sempre vicina in questi lunghi mesi trascorsi ad Hamburg e per avermi aiutato a superare momenti difficili con la sua presenza costante e per me insostituibile, dandomi così la forza di continuare e di giungere sino alla fine.

Queste persone resteranno nei miei ricordi. Grazie a tutti.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Prefazione

Con il trascorrere degli anni i mass-media sono divenuti un elemento estremamente rilevante nella nostra cultura, tanto che ora occupano buona parte del nostro tempo sia per motivi legati al lavoro, che per motivi legati allo studio od al semplice divertimento.

Ciò che sarebbe interessante proporre, in tale universo di informazione, è un cambiamento della modalità di comunicazione; ciò significa, in altre parole, aggiungere ai nostri mezzi di comunicazione la possibilità di una discussione "molti a molti" su larga scala [Lesk 1989].

Sino ad ora il meccanismo di diffusione dell'informazione su larga scala è stato principalmente un meccanismo di tipo "uno a molti".

Per secoli le tecnologie "uno a molti" dei mezzi di comunicazione hanno permesso ai proprietari dei canali di distribuzione dell'informazione di dire qualsiasi cosa senza la possibilità di una risposta da parte dell'ascoltatore che giocava, nel dialogo, un ruolo solamente passivo. Televisione, radio e giornali sono mezzi attraverso i quali gli editori ed i giornalisti fanno pervenire messaggi a milioni di persone, sicuri del fatto che questi milioni di persone non potranno effettuare alcuna replica ai concetti che stanno passivamente ricevendo. Questo meccanismo porta all'inganno di molte persone in molteplici situazioni [Glouberman 1996].

Chiunque infatti dovrebbe avere la possibilità, leggendo ciò che le altre persone dicono attraverso articoli, dibatti politici o per mezzo della pubblicità, di aggiungere propri commenti e considerazioni personali. Questi ultimi poi potrebbero essere seguiti da analoghe risposte, nell'istaurarsi di una comunicazione "attiva" tra persone che appartengono ad un determinato gruppo [Glouberman 1996].

Nella comunicazione "molti a molti" ogni concetto espresso da una certa parte è completamente aperto alla discussione ed alla critica da parte degli altri. La rete Internet è ormai divenuta un mezzo di comunicazione, ma il World Wide Web è comunque basato sul vecchio modello di comunicazione "uno a molti". Trasformare il Web in un mezzo di comunicazione "molti a molti" porterebbe sicuramente ad un drammatico impatto sullo sviluppo futuro dei nostri mass-media [Lesk 1997].

Pensando al World Wide Web si potrebbe avere la possibilità di aggiungere commenti ogni volta che si visualizza una pagina di un determinato documento. Si potrebbe anche avere la possibilità di leggere commenti precedentemente formulati da altre persone nella costruzione di un vero e proprio dibattito attraverso la rete. È questo un concetto diverso rispetto a quello dei *New-*

sGroups che sono realizzati attualmente sui *Servers WWW* e che sono anch'essi un meccanismo di comunicazione "molti a molti".

Attraverso i *NewsGroups* è possibile creare dei gruppi di persone che si scambiano informazioni e notizie relative a determinati argomenti o questioni. Un aspetto importante da mettere in evidenza riguarda il fatto che, mentre i *NewsGroups* sono fondamentalmente *subject-oriented* (orientati per tematiche di interesse), le annotazioni sono *document-oriented* ovvero raggruppate e gestite in base al documento a cui fanno riferimento.

Il concetto di annotazione presenta connotazioni diverse e riguarda principalmente la possibilità di effettuare un'elaborazione personale di documenti multimediali, arricchendone i contenuti e ponendo le basi per l'istaurarsi di un lavoro cooperativo tra gli utenti del sistema a supporto del meccanismo di annotazione.

In tale testo viene descritto un sistema a supporto del meccanismo di annotazione di documenti multimediali presenti sul *World Wide Web*; per **annotazione** si intende una manipolazione creativa del documento a cui si riferisce, attraverso la quale il lettore assume un ruolo attivo arricchendone il contenuto informativo. Sulla base di tali concetti generali è stato svolto il lavoro descritto nei diversi capitoli in cui si struttura il testo.

1.1 Motivazioni

I documenti digitali presentano caratteristiche migliori rispetto ai documenti cartacei per numerosi motivi; essi sono più facili da modificare, riprodurre, distribuire e ricercare. Tutti questi vantaggi hanno portato alla rapida evoluzione di documenti digitali anche se la carta ha mantenuto alcune caratteristiche e funzionalità che non sono ancora disponibili per i documenti digitali. Una delle più importanti funzionalità che ancor oggi la carta mantiene è la possibilità di fare annotazioni con estrema rapidità e semplicità.

Un'annotazione relativa ad un documento può essere pensata come un commento od una domanda relativi ai contenuti del documento stesso; le annotazioni possono inoltre riferirsi ad altre annotazioni in quanto esse stesse documenti dotati di un contenuto proprio[1].

L'annotazione può essere quindi considerata la chiave attraverso cui l'analista annota le proprie interpretazioni relative ad un certo documento; essa spesso è proprio il processo intermedio tra la lettura e la scrittura . Gli analisti in genere non prendono nota scrivendo le loro osservazioni su fogli separati bensì sul documento stesso sottolineandolo, evidenziandone le parti più importanti, scrivendo delle note ai margini e così via. Queste annotazioni incrementano notevolmente il contenuto del documento arricchendolo di concetti rilevanti. La carta è un mezzo attraverso cui si possono facilmente memorizzare annotazioni, diversamente a quanto potrebbe essere fatto con un documento digitale [Levy and Marshall 1995].

Le ragioni che giustificano l'annotazione di documenti, siano essi cartacei oppure digitali, sono molte e tra loro diverse; l'annotazione può essere effettuata semplicemente per mettere in evidenza alcuni aspetti del contenuto del documento ritenuti di particolare importanza. Oppure l'annotazione può rappresentare un commento a concetti espressi nel documento originale

[1] Questo è un principio che sta alla base della creazione di una struttura ricorsiva per l'annotazione.

con il conseguente arricchimento dei suoi contenuti. Una precisa ed ordinata descrizione delle motivazioni che spingono ad annotare documenti è riportata nel capitolo 2.

In tale testo viene descritto un sistema a supporto delle annotazioni digitali a documenti multimediali; da questo punto di osservazione le annotazioni possono riguardare una grande varietà di oggetti tra cui testo HTML, immagini, suoni ed ogni altro tipo di oggetto che può essere referenziato tramite un URL.

Un'ulteriore possibilità potrebbe essere quella di configurare le annotazioni come servizi disponibili con il documento a cui fanno riferimento. In tal caso l'URL identifica un servizio, ovvero una procedura o *routine* (ad esempio una *routine* di ricerca) che verrà eseguita in riferimento al documento oggetto di annotazione.

Inoltre per le annotazioni si possono definire anche dei diritti d'accesso, ovvero delle regole in base alle quali si possono stabilire degli spazi di visibilità delle annotazioni stesse nell'ambito del sistema in cui vengono gestite. Molti *Browsers* moderni supportano *Personal Annotations* associate ad ogni documento, create e leggibili solamente dall'utente del *Browser*. In aggiunta a *Personal Annotations* un documento potrebbe anche avere un certo numero di *Public Annotations* create e leggibili da chiunque, ed infine sono possibili anche *Group Annotations* create e leggibili dai membri di un gruppo.

Con il termine annotazione, come descritto in tale sezione, ci si riferisce quindi ad un insieme di manipolazioni del documento che trasformano il ruolo del lettore in un ruolo attivo che porta ad un arricchimento del contenuto del documento. Nel caso di una lettura non ricreativa il coinvolgimento attivo del lettore nel documento è essenziale al fine della comprensione ed assimilazione del suo contenuto.

Da tutto ciò si può quindi dedurre che le annotazioni sono meta-informazioni che servono ad arricchire il contenuto dei documenti a cui si riferiscono. Le annotazioni di documenti multimediali costituiscono un presupposto per la personalizzazione e per un lavoro cooperativo nelle biblioteche digitali ed in generale nei sistemi che trattano informazioni multimediali.

I concetti di personalizzazione e cooperazione sono alla base del lavoro realizzato ed hanno condizionato enormemente le scelte effettuate in fase di modellizzazione e progettazione del sistema. Da un punto di vista generale per personalizzazione si intende un arricchimento dei documenti con notizie ed accenni personali; per cooperazione si intende la possibilità di scambio di annotazioni ed informazioni relative a determinati documenti disponibili in uno spazio condiviso. Una spiegazione più approfondita di tali concetti è riportata nel capitolo 4.

Le annotazioni possono essere utilizzate per molteplici scopi di cui non tutti sono proposti con il termine annotazione. Gli usi comprendono risposte a documenti od ad altre annotazioni (creando in questo modo delle conversazioni tra più persone), valutazioni sul valore e sull'appropriatezza di un documento, creazione di conversazioni multiutente in uno spazio condiviso; altri possibili utilizzi alternativi di un sistema di annotazione sono riportati nel capitolo 2.

Un sistema in grado di supportare le annotazioni può essere considerato come un ambiente in cui le persone possono collaborare quando lavorano con un insieme di documenti che devono essere condivisi. Tale sistema permette ad un gruppo di persone di condividere un insieme di documenti e di fare commenti relativi a quei documenti che sono in comune con altri membri del gruppo.

Poter avere le annotazioni in una forma digitale significherebbe conferir loro tutti i benefici

di cui già godono i documenti digitali. In aggiunta la forma digitale potrebbe fornire la possibilità di pensare a forme di annotazione completamente nuove, come ad esempio le annotazioni dinamiche in aggiunta ai più tradizionali commenti passivi.

1.2 Principi generali alla base di un *Annotation Server*

La costruzione di un sistema informatico che supporta il meccanismo di annotazione a documenti multimediali richiede anzitutto la realizzazione di un componente che si occupi di tutti gli aspetti relativi alla gestione delle annotazioni: tale componente può essere definito con il termine *Annotation Server*.

Un *Annotation Server* è un *Server* che memorizza e fornisce annotazioni relative ad uno o più documenti; tale *Server* è la risorsa principale di annotazioni. In esso si possono identificare dei componenti attivi che rispettano il paradigma *Evento-Condizione-Azione*. Tali componenti seguono una logica di funzionamento propria delle regole attive: quando si verifica un *Evento*, se è soddisfatta una determinata *Condizione* viene eseguita dal sistema una specifica *Azione*. Questi concetti sono esposti in modo formale nel capitolo 5.

Le annotazioni possono essere eseguite tecnicamente in vari modi e possono essere supportate da sistemi con logiche di funzionamento profondamente diverse.

Il mio lavoro segue una precisa serie di passi, i quali vengono riflessi completamente nella sequenza dei capitoli principali in cui si struttura il testo e che verranno descritti nella prossima sezione.

Le principali fasi che si possono identificare sono le seguenti:

- Classificazione dei diversi approcci ed identificazione dei concetti basilari relativi ai sistemi che supportano un meccanismo di annotazione;
- Studio ed analisi dei modelli e delle architetture esistenti;
- Sviluppo di un modello e di un' architettura;
- Implementazione di un sistema prototipale per l'annotazione di documenti HTML, con la possibilità di annotare l'intero documento, porzioni di testo e/o immagini in esso contenute.

Presupposto fondamentale dell'intero lavoro è lo studio dei sistemi già esistenti che supportano il meccanismo di annotazione; attraverso l'analisi di tali sistemi è possibile comprenderne aspetti positivi e negativi al fine di prendere le decisioni migliori in fase di progettazione. Sulla base di tale analisi si può effettuare una classificazione dei vari approcci che si possono adottare nell'organizzazione di una struttura a supporto del meccanismo di annotazione (capitolo 2).

L'aspetto principale è quindi l'elaborazione di un modello e di un'architettura per il sistema che dovrà essere progettato; il modello proposto è realizzato ponendo grande attenzione ai principi di personalizzazione e di cooperazione (capitolo 4) che costituiscono la linea guida lungo la quale tutto il lavoro si sviluppa.

Per tali ragioni il sistema prevede l'organizzazione degli utenti in progetti di sviluppo dotati di obiettivi da raggiungere; i progetti a loro volta sono organizzati in una struttura gerarchica ed ogni progetto prevede la presenza di una figura (*Project Administrator*) che ha lo scopo di svolgere tutte le funzionalità di amministrazione del progetto stesso. Questi sono concetti che stanno alla base della realizzazione di una struttura che consente lo svolgimento di un lavoro cooperativo.

Il sistema offre inoltre la possibilità di svolgere annotazioni private, di gruppo oppure di pubblico interesse; per le annotazioni sono cioè definiti degli *Access Modifiers* che ne determinano le regole di visibilità nell'ambito della struttura gerarchica dei progetti. Altro aspetto interessante è rappresentato dai meccanismi di notificazione; il sistema realizza tali meccanismi attraverso la spedizione di *emails* agli utenti interessati al verificarsi di certi eventi; tali eventi sono la creazione di una nuova annotazione, la cancellazione di annotazioni esistenti, l'inserimento o la cancellazione di un utente da un gruppo di progetto.

Infine il sistema mette a disposizione dei criteri flessibili per la ricerca e visualizzazione delle annotazioni, meccanismi per la gestione degli utenti e funzionalità per l'organizzazione e la gestione dei progetti; il tutto è integrato da dei meccanismi di autentificazione degli utenti del tutto ortogonali al modello realizzato e quindi indipendenti dalle logiche di funzionamento del sistema[2].

L'implementazione riguarda principalmente la realizzazione dell'*Annotation Server* e della base di dati a supporto delle annotazioni; le applicazioni che fanno parte dell'*Annotation Server* sono realizzate come metodi remoti che possono essere richiamati da un qualsiasi *Client* facente parte del sistema. Sono inoltre realizzati tutti i meccanismi di comunicazione tra l'*Annotation Server* e la base di dati nonché tra l'*Annotation Server* ed il *Proxy Server*.

Questo testo presenta quindi una risposta concreta, tramite la proposta di un modello, di un'architettura nonché dell'implementazione di un prototipo, all'esigenza crescente di avere un sistema che supporti il meccanismo di annotazione digitale di documenti multimediali presenti in uno spazio condiviso. Tale funzionalità è ritenuta di notevole utilità pratica in quanto consente un grande arricchimento delle potenzialità dei documenti digitali colmando una delle più grosse lacune che questi presentano nei confronti dei documenti cartacei: la possibilità di avere un meccanismo di annotazione digitale semplice, veloce e soprattutto flessibile.

L'implementazione dei meccanismi di fusione tra il documento originale e le annotazioni che ad esso si riferiscono, meccanismi che vengono realizzati sul *Proxy Server*, e dell'interfaccia grafica realizzata sui *Browsers* sono oggetto di un altro lavoro di tesi in fase di sviluppo presso la Technische Universität Hamburg-Harburg.

1.3 Contenuti

In tale sezione verrà brevemente descritta la struttura generale del testo attraverso una sintetica analisi del contenuto di ogni capitolo.

[2]Ciò significa che è possibile modificare i meccanismi di autentificazione senza per questo portare delle modifiche alle modalità in base alle quali funziona l'intero sistema.

Il testo si struttura in tre parti che riflettono la trattazione dell'argomento a diversi e progressivi gradi di dettaglio:

I Annotazioni di documenti multimediali in spazi condivisi in tale parte vengono presentati in prima istanza i concetti generali che stanno alla base del principio di annotazione di documenti multimediali (capitolo 2). Quindi si procede con una breve descrizione dei modelli esistenti a supporto del meccanismo di annotazione (capitolo 3), per concludere con l'esposizione di alcuni principi fondamentali (Personalizzazione e Cooperazione) che contestualizzano il meccanismo di annotazione nell'ambito del campo di ricerca relativo alle Biblioteche Digitali (capitolo 4).

II Sistema di annotazione a supporto della cooperazione e personalizzazione in questa parte viene presentata una descrizione dettagliata degli aspetti esposti, a livello generale, nella parte precedente mostrando particolare attenzione ai principi di cooperazione e personalizzazione.

In primo luogo viene presentato un modello astratto per un sistema a supporto del meccanismo di annotazione di documenti multimediali in uno spazio condiviso (capitolo 5); segue una breve descrizione delle tecnologie e delle metodologie di sviluppo utilizzate (capitolo 6) per concludere con la presentazione dell'architettura progettata nonché delle specifiche funzionali del prototipo realizzato (capitolo 7).

III Annotation Server: implementazione, applicazioni e valutazioni in questa parte vengono trattati in dettaglio alcuni aspetti relativi all'implementazione del sistema di annotazione presentato (capitolo 8) ed inoltre vengono effettuate alcune considerazioni conclusive cercando di identificare i campi di applicazione del sistema di annotazione proposto ed i suoi possibili sviluppi futuri (capitolo 9).

Parte I

Annotazioni di documenti multimediali in spazi condivisi

Capitolo 2

Requisiti di un sistema per l'annotazione

La definizione delle caratteristiche principali, che deve possedere un sistema a supporto dell'annotazione di documenti multimediali e che derivano da una scrupolosa analisi dei requisiti richiesti dallo specifico dominio applicativo, costituisce un passo di fondamentale importanza preliminare alla procedura di progettazione di un nuovo sistema[1].

Ciò giustifica la scelta di dedicare un intero capitolo a tali aspetti cercando quindi, sulla base dei modelli già esistenti descritti nel capitolo 3, di comprendere a fondo i requisiti richiesti da un sistema destinato a supportare un meccanismo di annotazione.

Nelle sezioni che seguono sono elencati i requisiti identificati, i quali costituiscono la base di partenza per lo sviluppo del modello presentato nel capitolo 5 . Tali considerazioni, come si avrà occasione di comprendere durante la lettura, implicano inevitabilmente dei *trade-offs* che a loro volta comportano delle scelte. Le scelte effettuate non sono descritte all'interno di tale capitolo bensì in quello relativo al modello (capitolo 5) in cui sono riportate le motivazioni e le giustificazioni che hanno condotto a determinate decisioni.

In questo capitolo invece verranno analizzati[2] tutti gli aspetti che caratterizzano le annotazioni in modo tale da aver un'immagine chiara e ben definita dell'argomento in oggetto, al fine quindi di poter modellizzare un sistema che consenta la loro amministrazione, diffusione e rappresentazione.

2.1 Definizione di annotazione

Il primo passo da compiere nello studio dei requisiti e delle caratteristiche di un sistema a supporto dell'annotazione di documenti multimediali consiste nel formulare una definizione del concetto di annotazione.

[1] Ciò costituisce una fase di estrema importanza preliminare alla modellizzazione e che consente l'identificazione di quegli aspetti del sistema che dovranno essere considerati con maggiore attenzione e di quelli che invece potranno essere trascurati.

[2] L'analisi è stata effettuata sulla base dello studio dei sistemi di annotazione già esistenti e descritti brevemente nel capitolo 3.

Per **annotazione** si intende una qualsiasi meta-informazione associata ad un documento, la quale fornisce un arricchimento del contenuto informativo del documento medesimo.

Le annotazioni a documenti vengono tipicamente effettuate nel momento in cui il lettore assume un ruolo attivo nella lettura, elaborando i contenuti espressi nel testo originale e formulando considerazioni che arricchiscono le informazioni in esso riportate.

Quindi un'annotazione può essere definita come il completamento di un documento con osservazioni personali che vanno ad arricchire il suo contenuto. Le annotazioni possono essere viste sotto svariate prospettive ed analizzate prendendo in considerazione ad esempio cosa viene annotato, in cosa consistono le annotazioni, chi esegue le annotazioni, a cosa si riferiscono le annotazioni e così via.

Le sezioni che seguono hanno lo scopo di analizzare tutti i vari aspetti che scaturiscono dalla definizione del concetto di annotazione, prendendoli in considerazione ed analizzandoli singolarmente in modo tale da comprenderne le implicazioni e le correlazioni reciproche.

A ciò segue una descrizione delle caratteristiche che devono essere possedute da un sistema a supporto dell'annotazione digitale, ponendo particolare attenzione ai meccanismi di gestione delle annotazioni, alle architetture e tecnologie disponibili, ai principi di organizzazione logica degli utenti ed alle proprietà che caratterizzano il concetto di annotazione.

2.2 Cosa viene annotato

La prima domanda che occorre porsi è: cosa viene annotato, ovvero in cosa consistono praticamente gli oggetti e gli elementi a cui possono essere associate delle annotazioni? Questa domanda può avere molteplici risposte tra loro molto diverse.

In generale si potrebbe asserire che un'annotazione può essere riferita ad un documento accessibile tramite un qualsiasi *Browser Client* che si trova appeso alla rete. Questa in realtà è un'affermazione piuttosto generale che necessita ulteriori spiegazioni.

In dettaglio si possono identificare tre concetti che consentono di comprendere a fondo cosa costituisce oggetto di annotazione:

- Granularità
- Oggetti Multimediali
- Annotazione ricorsiva

Granularità

Un'annotazione potrebbe essere relativa all'intero documento che si trova sulla rete e che può essere referenziato con un suo URL oppure potrebbe essere collegata solamente ad una parte del documento stesso, come ad esempio ad un paragrafo, ad una frase o semplicemente ad una parola contenuta nel testo che si sta analizzando. Quest'ultima è la soluzione adottata, per esempio, nel modello Pan-Browser[3] (sezione 3.1). In tal caso l'utente ha la possibilità di referenziare non

[3]Questa è la soluzione adottata anche nel prototipo proposto in tale testo.

solo il documento a cui la sua annotazione si riferisce ma anche una porzione di testo interna al documento.

Ultima considerazione a tal riguardo è relativa al fatto che i documenti oggetto di annotazione digitale possono essere in formato HTML oppure in un qualsiasi altro formato quale ad esempio il formato *Word*, che ormai è divenuto uno standard di fatto nell'ambito dell'elaborazione di testi digitali.

Oggetti Multimediali

Un aspetto particolarmente interessante consiste nel fatto che i documenti presenti sul World Wide Web sono multimediali, ovvero comprendono, oltre al testo, anche altri elementi quali immagini e suoni.

Le annotazioni potrebbero essere riferite anche alle immagini. Ovviamente vi sono sistemi che ancora non supportano un meccanismo di annotazione delle immagini (ComMentor, sezione 3.3). Nonostante ciò questa funzionalità potrebbe essere molto utile in quanto l'utente potrebbe esprimere commenti e considerazioni personali riguardanti gli oggetti raffigurati nelle immagini che compaiono nel documento.

Si pensi a tal proposito al progetto WEL (*Warburg Electronic Library*, sezione 4.2.2) [Niederée *et al.* 1996] che ha l'obiettivo di costruire una biblioteca digitale in cui sarà contenuta una grossa quantità di immagini storiche. In un tale contesto un meccanismo che consenta di annotare anche le immagini è di grande utilità ad esempio per coloro che desiderano fare determinate ricerche annotando passo per passo i risultati ottenuti.

Annotazione ricorsiva

Le annotazioni, essendo esse stesse dei documenti referenziabili tramite un URL, potrebbero essere oggetto di annotazione. Ciò comporta la creazione di un meccanismo ricorsivo in cui si deve avere la possibilità di referenziare non solo i documenti di base ma anche le annotazioni che si riferiscono a tali documenti od ad oggetti interni a tali documenti[4].

La possibilità di poter fornire un supporto all'annotazione ricorsiva è molto interessante in quanto consentirebbe l'organizzazione di conversazioni tra utenti. Ad esempio un utente potrebbe esprimere un commento relativo al contenuto di un certo documento; a questo punto il proprietario del documento, letta l'annotazione effettuata, potrebbe decidere di rispondere con un ulteriore commento.

La cosa interessante di un tal meccanismo è che più utenti potrebbero fare annotazioni relative ad un certo documento, ed il proprietario, ad esempio, potrebbe rispondere ad ognuna di esse con delle argomentazioni diverse; ciò consentirebbe l'evolversi di vere e proprie conversazioni multiple.

[4]Come descritto nel capitolo 5, nel sistema sviluppato è possibile utilizzare un meccanismo di annotazione ricorsiva con la conseguente creazione di vere e proprie gerarchie di annotazioni.

2.3 Caratteristiche del meccanismo di annotazione

In questa sezione vengono prese in considerazione le proprietà che stanno alla base del meccanismo di annotazione di documenti multimediali. In primo luogo vengono analizzate le varie tipologie di annotazioni; segue quindi l'esposizione delle caratteristiche strutturali che configurano un'annotazione digitale, per concludere con la descrizione delle possibili modalità di utilizzo del meccanismo di annotazione.

2.3.1 Tipologie di annotazioni

Come riportato nella premessa a tale capitolo, un'annotazione può essere interpretata come una meta-informazione creata dall'utente. L'utente cioè deve avere la possibilità di creare, presentare e controllare le annotazioni che devono sempre essere riferite ad un oggetto referenziabile con un suo URL.

Sulla base di tali considerazioni a carattere generale si possono identificare tre diverse tipologie di annotazioni:

- Annotazioni Testuali
- Annotazioni Grafiche
- Annotazioni come Servizi

Annotazioni Testuali

Un'annotazione potrebbe consistere in un semplice commento, ovvero essere espressa in formato testuale ASCII oppure HTML, espressa sotto forma di piccole note riferite al documento stesso.

Questa modalità di annotazione digitale non risulta essere particolarmente complessa ed innovativa infatti rappresenta ciò che in realtà viene effettuato nella classica procedura di annotazione di documenti cartacei.

Di particolare interesse è la possibilità che l'utente possa fare liberamente delle annotazioni al documento utilizzando diversi strumenti, colori e simboli grafici. In tal caso il meccanismo di supporto dovrà essere più complesso e dovrà farsi carico, ad esempio facendo delle copie del documento originale, di preservare l'integrità del documento annotato.

Annotazioni Grafiche

Le annotazioni possono consistere anche in immagini, suoni ed in un qualsiasi oggetto referenziabile con un suo URL.

In base a tale logica le annotazioni possono consistere in sottolineature del testo con una striscia luminosa oppure in *hyperlinks* che rimandano a documenti testuali, immagini o suoni; ed ancora le annotazioni si possono presentare come delle marche poste ai margini della porzione di testo a cui si riferiscono, oppure come indicatori presenti in linea nel documento.

Ad esempio nel prototipo ComMentor (sezione 3.3) le annotazioni consistono in marche poste nella posizione del testo a cui l'annotazione si riferisce. Tali marche possono essere definite dall'utente e possono riportare varie immagini che indicano ad esempio l'autore oppure il gruppo di appartenenza.

Annotazioni come Servizi

Un aspetto estremamente interessante del modello NCSA (sezione 3.4) consiste nel fatto che le annotazioni potrebbero essere costituite anche da servizi correlati ai documenti a cui si riferiscono[5].

In altri termini in tal caso l'URL dell'annotazione referenzia un servizio; in tal modo è possibile associare dei servizi definiti dall'utente ad oggetti che costituiscono un qualsiasi documento HTML.

Da quanto detto quindi un'annotazione può consistere in una grande varietà di elementi; ovviamente in sede di progettazione si dovranno fare delle scelte in modo tale da sviluppare un modello che supporti la modalità di annotazione che più si adatta al contesto in cui tale meccanismo dovrà essere applicato.

2.3.2 Caratteristiche strutturali di un'annotazione

Per quanto riguarda le modalità in base alle quali organizzare le annotazioni il modello Multivalent Annotations (sezione 3.5) propone una classificazione estremamente interessante delle annotazioni in tre gruppi distinti in funzione delle loro caratteristiche strutturali:

- *Span Annotation*
- *Geometric Region Annotation*
- *Structural Annotation*

Span Annotation

Un primo gruppo è costituito dalle cosiddette *Span Annotations* che sono tutte quelle annotazioni che sono legate ad un intervallo di elementi, in genere caratteri, presenti nel testo. Tali annotazioni consentono di lavorare ad un livello di granularità molto fine nel senso che l'autore può riferirsi a precisi elementi che costituiscono il testo; tra queste annotazioni si possono comprendere le sottolineature con strisce luminose, gli *hyperlinks* ed i segni fatti sul testo utilizzando vari strumenti grafici a disposizione dell'utente.

[5]Per servizio si intende, dal punto di vista informatico, una *Routine* che viene mandata in esecuzione avendo come parametro di ingresso il testo o la porzione di testo a cui l'annotazione si riferisce.

Geometric Region Annotation

Il secondo gruppo è costituito dalle *Geometric Region Annotations* anche dette più semplicemente *Lenses*. Queste annotazioni possono essere considerate come dei semplici metodi (utilizzando la terminologia del paradigma *Object Oriented*) che vanno ad agire su oggetti che costituiscono il testo, mettendoli in evidenza.

Structural Annotation

Ultimo gruppo è costituito dalle *Structural Annotations* che sono tutte quelle annotazioni che si riferiscono ad un preciso elemento nella struttura del testo (es. capitolo, paragrafo, frase, etc...). Possono essere implementate anch'esse come metodi ovvero sotto forma di servizi come ad esempio il servizio di *cut and paste*.

2.3.3 Utilizzi alternativi del meccanismo di annotazione

Come asserito più volte le annotazioni possono essere considerate come meta-informazioni fornite dall'utente; il meccanismo di annotazione poi può essere sfruttato per la creazione e la gestione di altre strutture quali *trails, voting* e *seals of approval*[6].

Si possono individuare tre diverse modalità in base alle quali un sistema per l'annotazione di documenti multimediali potrebbe essere impiegato secondo logiche diverse dalla semplice annotazione:

- Meccanismi di votazione
- Marche posizionali
- Liste di ricerca

Meccanismi di votazione

Le annotazioni potrebbero essere utilizzate per esprimere un giudizio ed un voto relativo allo stile, al contenuto ed all'appropriatezza di un documento. Tale meccanismo è supportato dal modello Pan-Browser (sezione 3.1) il quale permette di raccogliere e conteggiare i voti relativi ad un documento quando questo viene caricato.

Inoltre le annotazioni possono essere utilizzate per realizzare *Seals Of Approval (SOAPs)*; SOAPs sono meta-informazioni contenenti stime su certi documenti che possono essere organizzati sotto forma di *Annotation Sets*.

Infine utilizzando il meccanismo di annotazione si possono realizzare i cosiddetti "indicatori d'uso" allo scopo di conteggiare il numero di utenti che accedono ad un determinato documento. In tal caso le annotazioni potrebbero essere organizzate in un *Annotation Set* chiamato *Usage Set* in cui il numero di annotazioni presenti in un determinato momento riflette il numero di utenti che hanno fatto accesso al documento referenziato da quel particolare URL.

[6]Queste sono modalità in base alle quali il meccanismo di annotazione viene utilizzato secondo logiche diverse da quelle che caratterizzano la classica procedura di annotazione.

Marche Posizionali

Altra modalità in base alla quale potrebbero essere sfruttate le annotazioni è quella delle *landmarks*. Le *landmarks* sono dei posti, nel Web, con i quali l'utente è familiare e che vuole avere come punti di riferimento. A tal scopo l'utente potrebbe annotare tali documenti con annotazioni che fanno parte tutte di uno stesso insieme; interrogando poi l'*Annotation Server* su quel particolare *Annotation Set* l'utente può facilmente ottenere tutti i documenti che desidera con estrema rapidità.

Le *landmarks* sono estremamente utili se accoppiate con il meccanismo del *tour*, realizzando una sorta di filtro collaborativo. Il meccanismo del *tour* consiste nell'organizzare una semplice lista di *links* ad annotazioni che guidano l'utente in base ad una logica ben definita attraverso una serie di documenti, dispersi sulla rete, senza alcun bisogno di cercarli esplicitamente.

Liste di ricerca

Le annotazioni potrebbero essere utilizzate anche per costruire *trails*, ovvero strutture in cui ogni elemento è costituito da un'annotazione, un puntatore all'elemento successivo ed un puntatore all'elemento precedente. In tal caso le *trails* potrebbero essere utilizzate per creare determinati percorsi logici attraverso una serie di documenti con la possibilità di annotare ciascuno di questi. In tale logica si potrebbe pensare anche di legare le annotazioni a specifiche versioni di un documento oppure anche creare dei meccanismi di correlazione tra le stesse.

Inoltre le *trails* possono essere applicate per implementare *tours* multipli guidati mediante la costruzione di *Annotation Sets* all'interno di ognuno dei quali le annotazioni vengono collegate in *tours* coerenti con la semantica del set.

Altra modalità di impiego delle annotazioni sono le *hotlists* condivise che non sono altro che *Annotation Sets* a cui hanno accesso più gruppi che condividono la *hotlist*. Ogni *Annotation Set* è un elemento della *hotlist*.

2.4 Principi di organizzazione delle annotazioni

Uno degli aspetti veramente importanti che si devono prendere in considerazione nel momento in cui si progetta un sistema a supporto delle annotazioni è quello relativo alle modalità in base alle quali le annotazioni vengono organizzate.

Nei modelli analizzati nel capitolo 3 un principio di fondo che è riscontrabile in ognuno di essi consiste nel fatto che le annotazioni vengono organizzate in *Annotation Sets* che risiedono su di un *Annotation Server* addibito appositamente alla gestione delle annotazioni.

Le annotazioni che appartengono ad uno stesso insieme possono essere trattate coerentemente come se fossero un unico documento in termini di mantenimento della versione, di controllo degli accessi, di archiviazione etc.

Ogni *Annotation Set* quindi, essendo situato su di un determinato *Server*, è identificato da un preciso URL; praticamente ogni *Annotation Set* potrebbe essere visto come un documento distribuito in quanto ogni annotazione che appartiene allo stesso insieme si riferisce a documenti diversi e situati in una qualsiasi parte del mondo.

Le annotazioni possono essere raggruppate sostanzialmente in base ai seguenti criteri:

Autore: in tal caso un *Annotation Set* è costituito da tutte quelle annotazioni che sono state create dalla stessa persona, indipendentemente dai documenti a cui si riferiscono e dal particolare gruppo di utenti a cui appartiene l'autore

Soggetto: ogni annotazione può avere un tema che sintetizza il suo contenuto informativo. Una possibile modalità di raggruppamento è quella di associare tutte le annotazioni che si riferiscono allo stesso tema

Documento: in tale logica si formano gruppi di annotazioni che si riferiscono tutte allo stesso documento

Gruppo di utenti: è possibile costruire *Annotation Sets* che comprendono tutte quelle annotazioni scritte da persone che appartengono ad un determinato gruppo di utenti. In tal caso il sistema a supporto dell'annotazione deve anche fornire un supporto all'organizzazione logica degli utenti (sezione 2.7).

Infine, ortogonalmente ai criteri di raggruppamento individuati, le annotazioni che risiedono su di uno stesso *Annotation Server* possono essere organizzate in un *tour*. Ciò significa che può essere creato un insieme costituito da tutte quelle annotazioni che sono state create più di recente e che quindi saranno richieste con una probabilità più elevata.

2.4.1 Modalità di accesso alle annotazioni

In linea di principio si possono pensare dei meccanismi in base ai quali gli *Annotation Sets* possono essere resi accessibili solo a determinati gruppi di utenti organizzati secondo modalità spiegate di seguito (sezione 2.7). In tale logica possono essere create anche annotazioni personali raggruppate in *Annotation Sets* non accessibili da nessuno.

In sostanza nell'ambito di un sistema per l'annotazione di documenti multimediali ci potrebbero essere **annotazioni di gruppo**, accessibili solo da determinati gruppi di utenti opportunamente predefiniti, **annotazioni pubbliche**, accessibili da chiunque indipendentemente dal gruppo di appartenenza, ed **annotazioni personali** accessibili solo dall'utente che le ha create.

Questi, a grandi linee, sono i principi fondamentali che possono essere seguiti nel momento in cui si deve decidere quale organizzazione logica dare alle annotazioni nel momento in cui si progetta un sistema a loro supporto.

2.5 Meccanismi di gestione delle annotazioni

Questa sezione riveste un ruolo di primaria importanza nell'ambito dell'intero capitolo in quanto in essa vengono descritti i possibili meccanismi di gestione delle annotazioni e dei documenti a cui esse si riferiscono; tali meccanismi sono stati individuati a seguito dell'analisi di modelli già esistenti e che supportano, in modi diversi, la procedura di annotazione di documenti multimediali (vedi capitolo 3).

In generale un meccanismo di gestione delle annotazioni deve consentire la fusione tra meta-informazioni distribuite e documenti a cui si riferiscono, fornendo in tal modo un importante supporto al concetto di personalizzazione.

Altro aspetto da non trascurare nel meccanismo di gestione è il prevedere la possibilità che ci possano essere annotazioni che si riferiscono ad altre annotazioni; in altre parole deve essere prevista una forma di annotazione ricorsiva.

Nel seguito di tale sezione vengono analizzate in dettaglio le principali operazioni attraverso le quali è possibile gestire la procedura di annotazione di documenti multimediali; le funzionalità descritte sono le seguenti:

- Visualizzazione
- *Editing*
- Meccanismi di *merge*
- Meccanismi di ricerca

Ogni operazione presenta particolari caratteristiche e può essere realizzata in base a principi e logiche diverse, le quali mettono maggiormente in evidenza determinati aspetti trascurandone inevitabilmente degli altri.

2.5.1 Visualizzazione

La visuallizzazione delle annotazioni è un processo che dipende fortemente da ciò che si intende per annotazione. Come analizzato nelle sezioni precedenti le annotazioni possono essere espresse sotto forma di testo, immagini, suoni e simboli grafici di vario tipo. In ognuno di tali casi le modalità adottate al fine della loro visualizzazione possono essere molteplici e molto diverse tra loro.

Inoltre occorre valutare anche la possibilità che l'utente voglia visualizzare solamente le annotazioni che non ha ancora letto, senza andare a ricercarle tra l'insieme di tutte le annotazioni che in un determinato momento sono riferite al documento originale; tale principio ha delle implicazioni rilevanti sia nella fase di modellizzazione che nella fase di implementazione del sistema.

Da un punto di vista generale si possono individuare due modalità in base alle quali visualizzare le annotazioni:

- Simboli grafici interni al documento
- Albero gerarchico delle annotazioni

Simboli grafici interni al documento

Nel caso in cui le annotazioni si riferiscono ad un certo documento nella sua globalità, esse potrebbero essere visualizzate sotto forma di semplice testo HTML.

Nel caso in cui invece le annotazioni si riferiscono a porzioni di testo interne al documento originale, esse potrebbero essere immerse nel testo ed identificate da piccole icone (come accade nel progetto ComMentor (sezione 3.3)). In tal caso la loro visualizzazione avviene semplicemente "cliccando" il bottone del *mouse* sopra l'icona relativa all'annotazione desiderata.

Albero gerarchico delle annotazioni

Altro criterio generale in base al quale visualizzare le annotazioni è quello presentato nel prototipo CoNote (sezione 3.2), nel quale le annotazioni che si riferiscono ad un determinato documento vengono visualizzate sotto forma di lista; tale lista può presentare elementi indentati nel momento in cui si hanno annotazioni che si riferiscono ad altre annotazioni piuttosto che al documento di base. Per visualizzare l'annotazione desiderata è sufficiente "cliccare" il bottone del mouse sulla stringa che la identifica.

Tale principio secondo cui le annotazioni vengono visualizzate in una logica indentata è adottato anche nel prototipo NCSA (sezione 3.4) e comporta la creazione di un vero e proprio albero gerarchico di annotazioni.

2.5.2 *Editing*

L'aspetto relativo all'*editing* delle annotazioni, ovvero alla possibilità di poter modificare le annotazioni, è una questione abbastanza delicata e che merita quindi molta attenzione. Infatti la modifica di annotazioni già effettuate potrebbe comportare problemi notevoli ad esempio nel momento in cui il sistema supporta un meccanismo di annotazione ricorsiva. In tal caso modificare il contenuto di un'annotazione potrebbe significare far perdere significato all'intera gerarchia che eventualmente si struttura al di sotto di essa.

In linea di principio si possono individuare le seguenti logiche in base alle quali organizzare l'*editing*:

- *Editing* non consentito
- Controllo degli accessi

Di seguito vengono brevemente descritti i principi che stanno alla base di tali criteri.

Editing non consentito

Anche l'*editing*, come del resto la visualizzazione, è un processo che dipende fortemente dalla tipologia di annotazioni supportate dal sistema. In fase di *design* comunque si potrebbe decidere di non dare alcuna possibilità di modifica delle annotazioni; questa è ad esempio la soluzione adottata nel modello ComMentor (sezione 3.3) in cui le annotazioni possono essere solo create oppure distrutte ma non modificate. Anche quella di non consentire l'*editing* delle annotazioni è quindi una scelta che deve essere opportunamente valutata nella fase di progettazione del sistema.

Controllo degli accessi

L'*editing* delle annotazioni assume una connotazione particolare nel momento in cui viene considerato come un processo correlato al meccanismo di autentificazione che verrà descritto nella sezione 2.7.3. Ciò significa che il sistema prevede la possibilità di fare *editing* delle annotazioni ma non tutti gli utenti che interrogano un documento possono avere il diritto di poter modificare le annotazioni ad esso relative.

In tal caso occorre quindi progettare un meccanismo di controllo degli accessi, il quale, secondo logiche predefinite, stabilisce ciò che l'utente può e non può fare in relazione alle annotazioni appese ad un determinato documento.

2.5.3 Meccanismi di *merge*

Il *merge* tra le annotazioni ed i documenti a cui queste si riferiscono è un meccanismo che può essere anche molto complesso e che dipende dalle decisioni che si sono prese riguardo le modalità di archiviazione delle annotazioni.

In linea di massima occorre individuare l'insieme di criteri in base ai quali organizzare i riferimenti alle annotazioni all'interno del documento originale al fine poi di poterle recuperare e posizionare correttamente nel momento in cui l'utente le richiede.

In secondo luogo occorre identificare quali sono i componenti architetturali a cui è delegata la responsabilità di tali meccanismi di *merge* tra documento ed annotazioni che ad esso si riferiscono.

Riferimenti per le annotazioni

Le procedure da definire per la fusione di documenti ed annotazioni sono particolarmente critiche nel momento in cui le annotazioni sono memorizzate su archivi separati da quelli dei documenti a cui si riferiscono. Ciò è quanto avviene nel sistema Pan-Browser (sezione 3.1) in cui i documenti non vengono in alcun modo preparati per essere annotati; in un tale contesto il meccanismo di sintesi tra documenti ed annotazioni è di fondamentale importanza. Esso consiste essenzialmente nell'inserire delle marche (**posizioni assolute**) interne al testo che non sono altro che degli *hyperlink* ad elementi di una lista appesa al documento. Ogni elemento di tale lista è un *link* ad un'annotazione che viene attivato dalla selezione di un opportuno bottone presentato attraverso un'apposita maschera di interfaccia.

Altra logica è quella proposta dal sistema ComMentor (sezione 3.3); il concetto nuovo che in tale progetto viene introdotto è quello di **inserzione incrementale**. Con tale termine si intende la possibilità offerta all'utente di inserire annotazioni in posizioni specifiche del testo e non necessariamente secondo un'ordine che segue l'evolversi del documento stesso. Questo principio, da come è stato presentato, consente una massima interattività e generalità ma introduce anche numerosi problemi a livello tecnico.

L'inserzione incrementale infatti comporta una modifica radicale dell'algoritmo di *merge* tradizionale; in base a tale nuova logica gli indicatori di posizione globale (che possono essere

pensati come semplici contatori) non possono più essere utilizzati e dovranno essere sostituiti con degli indicatori di posizione relativa, sicuramente più complessi e quindi più difficili da gestire.

Inoltre uno dei problemi che devono essere risolti nella definizione di un meccanismo di *merge* riguarda la modalità di gestione delle ridondanze; potrebbe infatti capitare che l'annotazione sia appesa ad una stringa che nel testo appare più volte. In tal caso il meccanismo di *merge* deve gestire tale ridondanza, ad esempio con *tags* interni al documento, in modo tale da poter posizionare correttamente l'annotazione.

Componenti architetturali responsabili del *merge*

In aggiunta a tutto ciò in sede di progetto occorre decidere anche quale deve essere l'entità destinata ad effettuare il *merge* tra annotazioni e documento. Le alternative principali possono essere tre: l'***Annotation Server***, il ***Client*** che richiede il documento oppure un componente esterno chiamato **Proxy Server**.

Da un punto di vista concettuale il meccanismo di *merge* è un compito proprio dell'*Annotation Server* il quale, sulla base di precise e ben determinate strutture dati, determina la posizione corretta delle annotazioni all'interno del documento effettuandone la fusione quando il documento viene richiesto dall'utente.

Nel secondo caso invece il *Browser* invia all'*Annotation Server* l'URL del documento di base, le informazioni di autentificazione relative all'identità dell'utente e l'indicazione di quale *Annotation Set* prendere in considerazione (questa ultima informazione ovviamente è necessaria nel momento in cui le annotazioni vengono organizzate in insiemi). Il *Browser* quindi, come risposta, riceve le annotazioni che rispettano i requisiti richiesti e le unisce al documento di partenza.

La terza alternativa consiste nel disporre di un apposito componente architetturale, il *Proxy Server*, specificatamente addibito all'esecuzione delle operazioni di reperimento e fusione di documenti ed annotazioni.

Queste considerazioni rappresentano, in breve, le questioni principali di un meccanismo, quale è quello di *merge*, che merita particolare attenzione in fase di progettazione del sistema.

2.5.4 Meccanismi di ricerca

La ricerca delle annotazioni che fanno riferimento ad un determinato documento può essere una procedura con una durata estremamente elevata se, nella progettazione del sistema, non si considerano a fondo i concetti che stanno alla base di tale meccanismo.

Scopo di tale sezione è presentare alcune delle soluzioni adottate in sistemi precedentemente sviluppati al fine di poter prendere la decisione migliore in fase di sviluppo del sistema, considerando congiuntamente anche le particolarità del dominio applicativo in cui il sistema stesso dovrà essere applicato.

In linea di principio occorre in primo luogo identificare le proprietà dell'algoritmo di ricerca e quindi descrivere meccanismi alternativi a supporto di tale algoritmo.

Proprietà dell'algoritmo di ricerca

In genere l'entità addibita all'esecuzione delle procedure di ricerca è l'*Annotation Server* in quanto tale componente può garantire risposte veloci essendo esso stesso responsabile della gestione delle annotazioni. Infatti una delle proprietà che l'algoritmo di ricerca deve possedere è proprio la **rapidità**

Altra importante proprietà che deve possedere un meccanismo di ricerca è la **flessibilità** ovvero la possibilità, offerta all'utente, di poter stabilire i più svariati criteri di ricerca costruiti in base a logiche personali. Tutto ciò può essere realizzato dando all'utente l'opportunità di definire liberamente numerosi parametri sulla base dei quali effettuare la ricerca delle annotazioni.

Il sistema AKI (sezione 3.6) presta particolare attenzione ai meccanismi di ricerca delle annotazioni. Il meccanismo di ricerca di tale sistema fa uso di liste di URL che si riferiscono ad annotazioni; inoltre il sistema offre la possibilità di generare liste di *bookmarks* attraverso cui l'utente può ritrovare rapidamente poche ed utili annotazioni.

Il prototipo Pan-Browser (sezione 3.1) fornisce un debole meccanismo di ricerca, il quale è realizzato fornendo all'utente maschere di selezione costruite in base a rigide e predeterminate logiche; in un tale contesto l'utente non può filtrare liberamente ed in modo flessibile le annotazioni in quanto legato a schemi di ricerca predefiniti e sviluppati facendo uso solamente delle funzionalità principali di una normale base di dati.

Meccanismi a supporto della ricerca

A supporto dei meccanismi di ricerca si potrebbero poi pensare altre soluzioni quali ad esempio la **notificazione**[7], che consiste nell'avvisare gli utenti interessati circa annotazioni relative a determinati documenti; questo potrebbe evitare la necessità di dover ricercare le annotazioni in quanto è il sistema che automaticamente le notifica all'utente. La notificazione quindi più che come un meccanismo alternativo alla ricerca potrebbe essere considerato come meccanismo a suo supporto.

Alternativamente alla notificazione vi è il **polling** che può anch'esso essere considerato un meccanismo a supporto della ricerca e che consiste essenzialmente in un'interrogazione attiva e periodica da parte del *Client*, il quale richiede all'*Annotation Server* eventuali nuove annotazioni associate ad un determinato documento.

2.6 Architetture e tecnologie

Le tecnologie e le architetture che possono essere scelte al fine di supportare un meccanismo di annotazione sono numerose. In tale scelta uno dei principi che deve essere chiaro è che la soluzione adottata deve essere il più possibile indipendente dai cambiamenti e dalle specializzazioni dei *Web Browsers* e dei *Web Servers*. Questi concetti sono estremamente importanti in quanto il sistema a supporto del meccanismo di annotazione a documenti multimediali deve essere un sistema multiutente e distribuito.

[7] Tale meccanismo nel sistema proposto è realizzato sull'*Annotation Server*.

In questa direzione occorre identificare quali sono i componenti principali che fanno parte di un sistema a supporto del meccanismo di annotazione:

- *Client*
- *Annotation Server*
- *Proxy Server*
- *Document Server*

Di seguito è riportata una breve descrizione di ciascun componente mediante la descrizione generale delle funzionalità svolte e dei principi che ne giustificano la presenza nell'ambito dell'intera architettura.

Client

Il *Client* rappresenta il punto di interazione tra l'utente ed il sistema. Sul *Client* in genere viene realizzata l'interfaccia utente; in linea di massima i principi che si devono perseguire nella realizzazione di un'interfaccia utente sono quelli di flessibilità ed indipendenza dalla piattaforma.

Ad esempio il prototipo ComMentor (sezione 3.3) presenta un'interfaccia utente sofisticata, che consente elevata flessibilità ma che non è standard e quindi non può funzionare su tutti i *Browsers*.

Inoltre potrebbe essere plausibile anche una soluzione che prevede la realizzazione del meccanismo di fusione tra documenti ed annotazioni proprio sul *Browser*. Questa soluzione consente di evitare inutili ricaricamenti di documenti, soprattutto se di grossa dimensione. In tal caso lo svantaggio è che non si possono sfruttare le eventuali maggiori funzionalità offerte dal proprio *Browser* in quanto il meccanismo di fusione deve essere standard.

Dal punto di vista delle tecnologie disponibili se si desidera avere estendibilità del sistema sono possibili diversi approcci quali ad esempio HTML più JAVA-applets, OpenDoc e OLE [Brockschmidt 1995]; queste sono scelte di tecnologie che garantiscono estendibilità ma consentono di lavorare ad un livello grossolano di granularità. Adobe's Acrobate ha invece definito un proprio formato noto come PDF che però, diversamente da HTML, è difficile da costruire. Microcosm [Fountain *et al.* 1990] è un sistema ipertestuale costruito sopra Microsoft Windows. Knowledge Weasel [Lawton and Smith 1993] sfrutta i vantaggi di tools come TCL e TK.

Annotation Server

L'*Annotation Server* è un componente fondamentale nell'ambito dell'architettura a supporto del meccanismo di annotazione; esso si occupa di gestire la base di dati contenente tutte le informazioni relative alle annotazioni e mantiene le indicazioni necessarie a reperire i documenti a cui esse si riferiscono.

Ad esempio, nell'architettura del sistema AKI (sezione 3.6) la base di dati è gestita dall'*Annotation Server* che può essere considerato come un *Gateway* tra *Browser* e base di dati stessa.

Altra possibilità potrebbe essere quella di avere non solo annotazioni e documenti separati ma anche un *Annotation Server* che mantenga solamente i riferimenti (ad es. attraverso gli URL) alle annotazioni senza immagazzinare i loro contenuti. Ciò consente una gestione più veloce del *Database* contenente soltanto i riferimenti delle annotazioni.

Altro aspetto che merita attenzione, ed esposto nella sezione 2.7.3, è quello relativo alla possibilità che le annotazioni siano private, pubbliche o di gruppo (*Access Modifiers* per le annotazioni). In tal caso si potrebbe pensare di avere un *Annotation Server* che si occupi della gestione delle annotazioni pubbliche ed un *Annotation Server* che gestisce le annotazioni di gruppo.

Proxy Server

Il *Proxy Server* rappresenta un componente addibito specificatamente alla realizzazione dei meccanismi di *merge* tra le annotazioni ed i documenti a cui esse si riferiscono.

Tale soluzione consente la realizzazione di meccanismi di *merge* molto sofisticati anche se introduce una maggior lentezza ripetto ai casi in cui la fusione tra documenti ed annotazioni avvenga a livello di *Browser Client* o di *Annotation Server*.

Document Server

Tale componente rappresenta l'unità architetturale di gestione di tutte le informazioni relative ai documenti annotati; in tal caso le annotazioni vengono gestite dall'*Annotation Server*, i documenti dal *Document Server* e la loro fusione può essere realizzata a livello di *Proxy Server*.

Questa soluzione è adottata ad esempio dal prototipo ComMentor (sezione 3.3) in cui le meta-informazioni (di cui le annotazioni ne sono un esempio) sono archiviate in una base di dati completamente separata da quelle che contengono i documenti. In tale modello infatti vi sono *Servers* che si occupano della gestione dei documenti (*Document Servers*) e *Servers* che invece si occupano della gestione delle meta-informazioni (*Annotation Servers*).

In alternativa ai *Document Servers* si potrebbe pensare di utilizzare direttamente l'infrastruttura del *World Wide Web* attraverso cui accedere all'immensa quantità di documenti gestiti dai *Servers Web* dispersi in tutto il modo, per poi annotarli utilizzando i meccanismi messi a disposizione dallo specifico sistema a supporto dell'annotazione.

In conclusione a questa sezione si può quindi affermare che le tecnologie e le architetture che si possono scegliere a supporto di un meccanismo di annotazione sono estremamente varie; la scelta comporta sempre un compromesso tra la possibilità di avere meccanismi flessibili ma non standard e viceversa.

In sede di progetto occorre quindi prendere le decisioni che sembrano più coerenti con le politiche globalmente adottate, accettando dei compromessi inevitabili come accade ogni volta che si è di fronte a meccanismi di *trade-off* come in questo caso.

2.7 Criteri di organizzazione logica degli utenti

Per organizzazione logica degli utenti si intende l'insieme di tutti quei principi e criteri in base ai quali gli utenti vengono classificati, suddivisi ed in generale gestiti dal sistema di annotazione.

Organizzare logicamente gli utenti costituisce un fattore estremamente importante al fine della realizzazione di una struttura cooperativa, nell'ambito della quale gli utenti possono lavorare in gruppo ottenendo buoni risultati in breve tempo.

2.7.1 Gruppi di utenti

Un modo per organizzare gli utenti, adottato ad esempio nel progetto Pan-Browser (sezione 3.1), è quello di suddividerli in gruppi all'interno dei quali gli utenti creano e condividono commenti relativi a documenti WWW. Il concetto di gruppo è quindi molto importante in un tale approccio ed ha delle implicazioni anche sul meccanismo di autentificazione che è analizzato di seguito (sezione 2.7.3). Ogni utente può appartenere ad uno o più gruppi; i gruppi possono essere privati oppure pubblici, ovvero possono essere aperti ad ogni utente oppure consentire l'accesso solo a quegli utenti che possiedono determinati requisiti[8].

Lo stesso concetto di gruppo è introdotto anche in ComMentor (sezione 3.3) ed in CoNote (sezione 3.2); nel sistema ComMentor ogni gruppo può essere associato ad uno o più *Annotation Sets*. Il meccanismo di creazione di un gruppo fa uso di maschere HTML e di funzionalità svolte dal *Browser*; l'utente stesso guida il processo di entrata a far parte di un gruppo e di creazione degli *Annotation Sets*.

Anche nel modello AKI (sezione 3.6) gli utenti sono organizzati in gruppi; gli interessi del gruppo sono gestiti da un amministratore dei gruppi; gli utenti possono appartenere anche a più gruppi i quali sono organizzati in base ad una struttura gerarchica. I gruppi inoltre possono essere aperti a tutti oppure essere dei gruppi privati a cui possono appartenere solo utenti con determinate caratteristiche.

2.7.2 Ruoli e diritti degli utenti

Oltre all'organizzazione degli utenti in gruppi un sistema per l'annotazione può prevedere anche la definizione di ruoli e quindi implicitamente di diritti legati ad ogni specifico ruolo. Ad esempio in Pan-Browser (sezione 3.1) gli utenti che fanno parte di un gruppo possono avere il diritto di accedere ad uno o più *Annotation Sets*; ogni utente ha il diritto di appartenere ai gruppi pubblici.

In CoNote (sezione 3.2) viene definito un chiaro e preciso insieme di ruoli che possono essere assunti dagli utenti; i ruoli possibili sono **viewer, reader, user** ed **author**. Ogni ruolo ha determinati diritti sulle annotazioni. Un *viewer* può visualizzare un documento annotato ma non può vedere le annotazioni che ad esso si riferiscono; un *reader* può leggere sia le annotazioni che il documento ma non può crearne delle nuove; un *user* può leggere ed anche aggiungere annotazioni; ed infine un *author* è un utente che ha tutti i diritti ovvero può leggere, aggiungere e cancellare le annotazioni.

Riassumendo, un buon meccanismo per l'annotazione deve prevedere anche una corretta definizione dei ruoli che gli utenti possono assumere e dei diritti che competono ad ogni ruolo in

[8] Per l'organizzazione degli utenti del sistema di annotazione si è adottata una logica analoga anche nel prototipo realizzato e proposto in tale testo.

modo tale da poter costruire, sulla base di ciò, un completo meccanismo di autentificazione che consente di offrire un sistema finale robusto e sicuro.

2.7.3 Meccanismi di autentificazione e di controllo degli accessi

I meccanismi di autentificazione e di controllo degli accessi possono essere svariati e dipendono fortemente dalle modalità adottate al fine dell'organizzazione degli utenti (sezione 2.7) e delle annotazioni (sezione 2.4).

In generale i principi che stanno alla base di un meccanismo di autentificazione sono i seguenti:

- Controlli per gruppi di utenti
- Controlli per gruppi di annotazioni
- Modalità di autentificazione

Controlli per gruppi di utenti

In genere se il sistema supporta gruppi privati e pubblici occorre avere un meccanismo che controlli l'accesso ai gruppi privati, ad esempio attraverso un meccanismo di validazione degli utenti.

In tale logica il sistema controlla l'identità dell'utente verificando la sua appartenenza ad un determinato gruppo.

Controlli per gruppi di annotazioni

Altra logica è quella di realizzare il meccanismo di autentificazione basandolo sugli insiemi di annotazioni.

Ad esempio nel prototipo ComMentor (sezione 3.3) il controllo degli accessi viene eseguito per *Annotation Sets* che vengono distinti in **privati**, a cui ha accesso una sola persona, **di gruppo** a cui ha accesso un gruppo ristretto di utenti e **pubblici** a cui hanno accesso tutte le persone. In tal modo il meccanismo di autentificazione è ortogonale al modello, ovvero ogni meccanismo può essere adottato indipendentemente dal sistema sottostante. Gli *Annotation Sets* a cui può accedere un determinato utente sono associati al suo profilo.

Modalità di autentificazione

Le modalità in base alle quali realizzare il meccanismo di autentificazione sono molte e dipendono dalla particolare implementazione del sistema a supporto dell'annotazione a cui devono essere applicate.

Il prototipo CoNote (sezione 3.2) supporta un meccanismo di autentificazione tradizionale attraverso cui l'utente richiede un documento inserendo il suo nome e la sua *password* che valgono anche per tutti i *links* successivi al documento.

Nel sistema NCSA (sezione 3.4) vi è addirittura il concetto di *Group Server* a cui l'utente si deve autentificare per accedere alle annotazioni di un determinato gruppo.

Dalle considerazioni fatte si può quindi dedurre che ci sono molti meccanismi a supporto dell'autentificazione; anche in tal caso la scelta deve essere effettuata in relazione alle politiche generali che si è deciso di seguire in fase di progetto cercando di adottare la soluzione che meglio si accorda con esse.

2.8 Requisiti di un *Annotation System*

A conclusione di tale capitolo è bene sintetizzare brevemente quali sono i requisiti principali richiesti ad un sistema a supporto dell'annotazione di documenti multimediali.

Tale sezione analizza in primo luogo quali sono le funzionalità principali che gli utenti richiedono ad un sistema per l'annotazione di documenti multimediali; sulla base di questa analisi segue una breve descrizione delle proprietà che dovrebbe avere un sistema a supporto dell'annotazione al fine di soddisfare le richieste degli utenti.

2.8.1 Richieste degli utenti

In realtà le richieste degli utenti dipendono dallo specifico dominio applicativo in cui verrà inserito il sistema finale funzionante. Le scelte fatte nel caso specifico del modello sviluppato e descritto in tale testo si inseriscono nel contesto del progetto WEL (*Warburg Electronic Library*, sezione 4.2.2) e quindi cercano di soddisfare appieno le richieste espresse dagli utenti finali di tale sistema.

In generale ciò che si richiede ad un sistema di annotazione è di **non dipendere dalle caratteristiche di uno specifico *Browser***; questo è un aspetto molto importante e che non può essere trascurato nella realizzazione di un nuovo sistema.

Inoltre gli utenti, in genere, quando pensano alla possibilità di fare annotazioni vogliono avere a disposizione un meccanismo di **comunicazione asincrona**[9].

Altra richiesta generale è quella che il sistema consenta una naturale **discussione di gruppo** (aspetto questo che non viene realizzato ad esempio attraverso sistemi di posta elettronica oppure meccanismi di *NewsGroups*). Più specificatamente potrebbe essere richiesta, ad esempio, la possibilità di effettuare commenti condivisi, filtri collaborativi, poter avere segnali di approvazione, tracce ed indicatori d'uso.

Altra caratteristica che in genere viene richiesta ad un sistema di annotazione è la **scalabilità** ovvero la possibilità che il sistema funzioni anche se alcune variabili vengono definite su di una più ampia scala. La scalabilità è una proprietà molto importante in un sistema di annotazione in quanto il numero di annotazioni che si riferiscono ad un determinato documento può crescere molto rapidamente determinando così la necessità di una riconfigurazione del sistema in un contesto più ampio.

[9] In base a tale criterio è stato pensato e realizzato il meccanismo di notificazione *off-line* descritto nel capitolo 5.

A ciò si aggiungono poi le richieste specifiche di ogni committente che ovviamente possono essere le più svariate e che possono fare riferimento a tutti gli aspetti analizzati nel corso di tale capitolo.

2.8.2 Caratteristiche di un sistema a supporto dell'annotazione

Per concludere vengono elencate alcune delle proprietà fondamentali che le annotazioni ed i sistemi a loro supporto devono possedere al fine di poter essere considerati tali.

In primo luogo le annotazioni devono essere riferite ad un documento od ad oggetti interni al documento e quindi devono apparire nelle **posizioni specifiche** dove si trovano gli elementi a cui si riferiscono. In secondo luogo le annotazioni devono essere dotate di **elevata espressività** nel senso che attraverso il meccanismo di annotazione l'utente deve essere in grado di esprimere ciò che realmente desidera.

Le annotazioni poi devono essere **indipendenti dal formato** del documento in modo tale da poter essere visualizzate anche con *Browsers* diversi da quello con cui sono state create; inoltre l'indipendenza dal formato del documento consente la realizzazione di un principio molto importante ovvero il fatto che le annotazioni rimangono appese al documento anche a seguito di cambiamenti nel suo formato.

Le annotazioni poi devono essere **estendibili** e **componibili**[10] con altre annotazioni; anche questa è una proprietà molto importante in quanto consente elevata flessibilità dell'intero meccanismo.

In aggiunta a tutto ciò il sistema a supporto delle annotazioni deve essere un sistema **distribuito** ed **aperto** ma soprattutto **indipendente dalla piattaforma** su cui è stato realizzato il documento. Questi principi sono di estrema importanza quando si utilizza un'infrastruttura distribuita su una larga scala quale è il *World Wide Web*.

In ultimo, ma non per questo meno importante rispetto alle altre proprietà, le informazioni devono essere **robuste** ovvero il meccanismo deve contemplare la possibilità che documenti ed annotazioni stesse possono cambiare in maniera del tutto asincrona senza per questo dar luogo a dei problemi di allineamento o di qualsiasi altro tipo.

Le annotazioni poi sono costituite da molti attributi alcuni dei quali sono obbligatori mentre altri sono facoltativi; tra gli **attributi obbligatori** vi deve essere l'autore, l'indirizzo, la data di produzione, la durata di validità ed un titolo pragmatico.

Ci possono essere poi altri **attributi opzionali** tra cui il tipo di annotazione (ovvero se si tratta di un commento, di un'informazione aggiuntiva, di una nota riassuntiva etc.), l'espressione di una valutazione ed infine un commento contenete osservazioni particolari.

In tale capitolo sono esposti i concetti pricipali sulla base dei quali si articola il resto del testo e soprattutto si struttura il modello realizzato e descritto nel capitolo 5.

[10] Con questi termini si intende la possibilità di realizzare annotazioni che si riferiscono ad altre annotazioni.

Capitolo 3

Modelli e Sistemi per l'annotazione

Le considerazioni riportate sistematicamente nel capitolo 2 consentono di giungere alla conclusione che con il termine di annotazione ci si può riferire a numerosi concetti estremamente diversi e con caratteristiche a volte contrastanti tra loro. Attraverso l'annotazione il lettore assume un ruolo attivo che lo porta ad assimilare a fondo i concetti riportati nel documento ed anche ad arricchire il contenuto del documento stesso. L'annotazione potrebbe essere considerata anche come un presupposto per concetti di più ampia portata quali sono quelli di **personalizzazione** e di **cooperazione**.

Attualmente esistono alcuni sistemi che sono stati sviluppati per supportare il meccanismo di annotazione e che possono essere classificati nel seguente modo:

- **Sistemi a supporto della discussione:** questi sistemi possono essere considerati come un naturale sviluppo dei *news-systems*. Esempi di tali sistemi sono HyperNews [D.LaLiberte 1997], [D.LaLiberte 1997], LineWire [MotherJones 1997] oppure CoNote [J.Davis 1996].

- **Sistemi multi-attore:** questo gruppo comprende tutti quei sistemi che consentono la produzione di documenti, separata sia nello spazio che nel tempo, da parte di molti attori; esempi di tali sistemi sono Quilt [Fish *et al.* 1988], IRIS [Sedlmayer 1996] oppure Collaborative Annotator [Koszarek *et al.* 1990].

- **Sistemi con spazio di lavoro condiviso:** questi non sono dei veri e propri sistemi a supporto del meccanismo di annotazione, bensì sono sistemi che consentono la costruzione di un documento da parte di gruppi di persone. Per quanto riguarda l'annotazione l'aspetto interessante è che tali sistemi rendono possibile la comunicazione tra membri del gruppo; esempi sono BSCW [R.Bentley *et al.* 1995] oppure HyperWave [GmbH 1997]. Il prototipo BSCW inoltre prevede opzioni attraverso le quali è possibile realizzare un rudimentale meccanismo di annotazione.

- **Sistemi generali per l'annotazione:** tali sistemi consentono di aggiungere informazioni ausiliarie ad un qualsivoglia documento residente sulla rete Internet. Questi sono dei veri e propri sistemi di annotazione a differenza di quelli appartenenti ai primi tre gruppi che forniscono solo delle funzioni aggiuntive a supporto di un tale meccanismo; esempi sono Ariadne [Konsortium 1996] oppure ComMentor [Roscheisen *et al.* 1997].

Questo capitolo riporta l'analisi di alcuni attuali sistemi di annotazione cercando di coglierne le differenze e capirne le logiche di funzionamento che stanno alla loro base. Questo studio costituisce il presupposto alla classificazione dei vari approcci e dei vari criteri che è riportata nel capitolo 2.

3.1 Pan-Browser

Il sistema *Pan-Browser* rappresenta un approccio per consentire ai membri di un gruppo di creare commenti indipendenti e condivisi relativi a documenti accessibili attraverso il Web. In particolare, il sistema supporta la creazione, la presentazione, la visualizzazione ed il controllo di meta-informazioni create dall'utente, le quali sono visualizzate con il corrispondente documento ma salvate indipendentemente da esso.

La funzionalità tipica di tali meccanismi è quella di supportare annotazioni relative a documenti accessibili attraverso *Web Browsers* appesi alla rete. Come proposto da [Röscheisen *et al.* 1995a], le meta-informazioni create dall'utente possono essere usate in modi diversi dalle annotazioni, modi questi che supportano la nozione di controllo editoriale (es. *voting, trails, seals-of-approval*, etc.).

Un aspetto importante di tale sistema riguarda il fatto che le annotazioni possono essere attaccate al termine del documento oppure possono essere riferite ad una particolare sezione del documento specificata dall'utente. Il vantaggio di tale approccio consiste nel fatto che esso offre molte possibilità di scalabilità ma sfortunatamente l'interfaccia utente e le funzionalità di interrogazione e sintesi sono strettamente correlate alla particolare implementazione del *Browser* ed ai meccanismi adottati per l'individuazione della posizione dell'annotazione all'interno del documento a cui si riferisce.

3.1.1 Architettura di riferimento ed implementazione

La figura 3.1 mostra a grandi linee l'architettura di riferimento mentre la figura 3.2 mostra più in dettaglio l'architettura *Strand* (*Stream TRANsducing Deamons*) [Brooks *et al.* 1995], [M.B.Davidson *et al.* 1995].

Nel modello descritto sinteticamente dalle figure riportate un utente singolo può essere membro di uno solo o di molti gruppi.

Ogni annotazione è membro di un preciso *Annotation Set*, ed ogni insieme di annotazioni risiede su di uno specifico *Annotation Server*.

Le annotazioni vengono raggruppate assieme in base a determinati criteri ed in genere un insieme comprende annotazioni tutte della stessa area.

In base a tale logica gli *Annotation Sets* forniscono all'utente un primo criterio di ricerca delle annotazioni che a lui più interessano. Inoltre ad ogni gruppo di utenti deve essere concesso l'accesso ad uno o più *Annotation Sets*. In tale modello non è importante il luogo in cui le annotazioni sono salvate bensì il meccanismo di memorizzazione e di reperimento delle annotazioni utilizzato dal GrAnT.

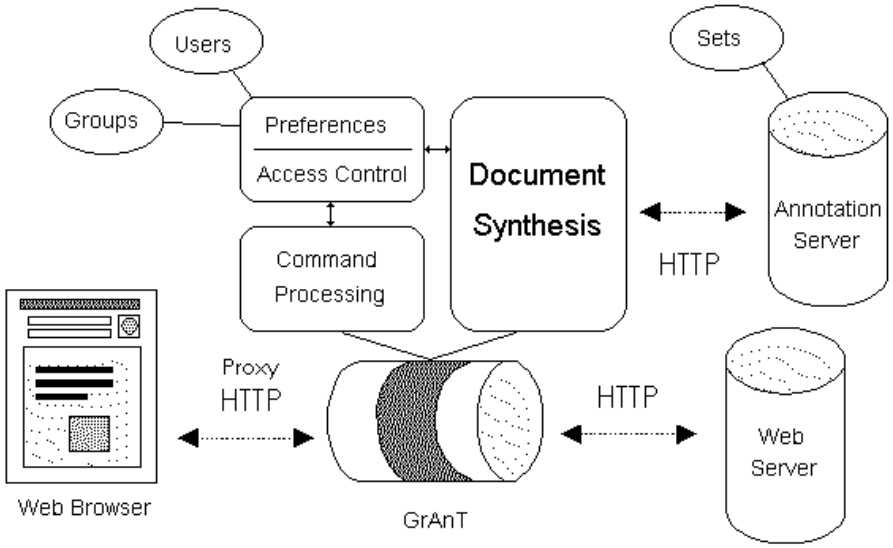

Figura 3.1: Architettura di riferimento.

Il meccanismo adottato è il protocollo HTTP [Berners-Lee and Connolly 1993] mentre il mezzo di memorizzazione è una versione largamente diffusa dell' *Annotation Server* sviluppato presso l'università di Stanford.

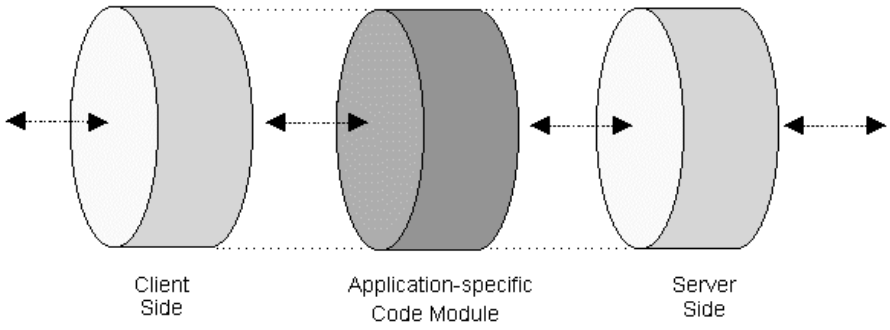

Figura 3.2: Architettura *Strand*.

3.1.2 Strand

Strand sta per *"Stream TRANsducing Deamons"*. Come si può vedere dalla figura 3.2 l'insieme di strumenti sviluppato fornisce un "involucro esterno" il quale si occupa di gestire la connessione tra *Client* e *Server*, tra i quali è posizionato il modulo codice di ogni specifica applicazione.

Il programmatore, grazie a questa struttura, può utilizzare un qualsiasi programma per lo sviluppo di sistemi al fine di creare il modulo codice dello *Strand*. Lo *Strand* assicura che il modulo sia connesso in un flusso di richiesta e risposta; in tal modo lo sviluppatore può operare tranquillamente nel suo contesto di sviluppo ignorando completamente le questioni relative alla rete. Lo svantaggio di una tale struttura consiste nel fatto che lo *Strand* aggiunge un ritardo al flusso di richiesta/risposta attraverso il protocollo HTTP.

L'interfaccia per interagire con l'*Annotation Service* è implementata attraverso una serie di maschere HTML e di controlli.

I componenti di interfaccia supportano due tipi di funzionalità principali ovvero la creazione di annotazioni ed il cambiamento dei profili[1].

Il GrAnT aggiunge bottoni di controllo allo sfonfo di ogni risposta HTML valida prima di inoltrare tale maschera al *Client* che l'ha richiesta. Questi bottoni includono le funzionalità di creazione di una nuova annotazione, di visualizzazione dei gruppi di utenti e di visualizzazione degli insiemi di annotazioni.

Il GrAnT cattura la risposta HTTP e guida l'utente in una serie di interazioni attraverso maschere HTML.

3.1.3 Flusso di controllo

Quando una richiesta HTTP raggiunge il GrAnT esso esegue una serie di passi in sequenza; in primo luogo crea un *thread di controllo* per manipolare l'intera transazione, quindi inoltra la richiesta al *Server* specificato dall'URL.

Il passo successivo consiste nell'autentificazione dell'utente e nella determinazione del suo profilo al fine di poter determinare quali annotazioni richiedere di seguito; fatto ciò si autentifica esso stesso all'*Annotation Server* e lo interroga su ogni annotazione associata con il richiesto URL e con il richiesto utente.

Il passo finale consiste nel sintetizzare il documento richiesto con le annotazioni ad esso associate e nel ritornare il tutto all'utente.

Per quanto riguarda l'interfaccia utente, il GrAnT intrappola le richieste che corrispondono a direttive di comando impartite dall'utente attraverso la selezione degli appositi bottoni di comando. Per esempio quando l'utente seleziona il bottone *"View Groups"*, una direttiva di comando *view groups* è inviata al GrAnT. Il modulo processa quindi la direttiva attraverso la spedizione di una maschera HTML di risposta al *Client*. Questa maschera include una lista di gruppi, lo stato corrente dell'utente in ognuno di tali gruppi ed un insieme di bottoni di comunicazione associati ad ogni gruppo.

Per inserire un'annotazione la maschera HTML che compare è pensata specificatamente per scrivere le annotazioni. Questa pagina prevede infatti una serie di campi di inserimento e di bottoni di comunicazione che possono essere utilizzati per l'inserimento di annotazioni. L'aspetto progettuale di rilevante importanza consiste nel fatto che le annotazioni possono essere appese

[1]Collezione di informazioni relative ad una persona. Il profilo utente può essere salvato nel *file system* di un *Browser* locale oppure remotamente su di un *User-Profile Server*.

ad una specifica porzione di testo (identificata per mezzo della pagina HTML a cui appartiene) oppure all'intero documento.

3.1.4 Considerazioni e problemi relativi al prototipo

L'interfaccia utente utilizzata è la GUI (*Graphical User Interface*) la quale sfrutta molte delle funzionalità delle maschere HTML; essa è potente così come semplice e soprattutto standard per tutti i *Browsers*. Lo svantaggio è che la barra degli strumenti e le annotazioni prevedono troppo spazio sulla pagina. Altro problema è la selezione del punto di attacco delle annotazioni all'interno del documento. Un naturale metodo di selezione consiste nell'indicare il punto di attacco con il *mouse* e nel selezionare un pezzo di testo che è inizialmente visualizzato dal *Browser*. Ora, siccome l'approccio deve essere *Browser-Independed*, l'accesso ad informazioni come gli eventi provocati dal *mouse* o dai *buffers* interni è ristretto.

Coerentemente a tale principio il prototipo implementa una semplice maschera HTML la quale inevitabilmente aggiunge una serie di passi noiosi al processo di selezione del punto di aggancio.

Invece, per una selezione diretta, l'utente deve fare *cut and paste* della porzione di testo di riferimento nel campo a ciò oppurtunamente addibito e fornito dalla maschera. A seguito di ciò il GrAnT esegue una serie di processi aventi lo scopo di individuare eventuali ridondanze presenti nel testo selezionato. Molti problemi si hanno nel momento in cui la stringa selezionata non è unica nel documento. Tale caso può essere gestito o rigettando l'annotazione oppure facendo in modo che l'utente selezioni la corretta occorrenza tra una lista di frasi ambigue.

Il prototipo qui presentato usa l'interfaccia CGI per il caricamento ed il recupero delle annotazioni. Questo approccio introduce una notevole ridondanza in un gran numero di vie. Ma le prestazioni dell'*Annotation Server* non sono l'unico problema. L'*Annotation Server* è abbastanza limitato in quanto offre solamente primitive funzionalità dei *Database* tradizionali.

Un *Annotation Server* dovrebbe consentire all'utente di filtrare e ricercare le annotazioni in base a criteri il più possibile flessibili come ad esempio in base al nome dell'autore, la data di creazione, il tipo di annotazione fornendo anche chiavi di ricerca sia per l'*header* che per il *body* dell'annotazione. Inoltre nel prototipo descritto ogni GrAnT assume l'esistenza di un singolo gruppo di lavoro che usa un unico *Database* locale per le annotazioni. Non sono ancora previsti meccanismi per la condivisione tra più gruppi di lavoro.

Altro problema è rappresentato dal fatto che l'attuale specifica HTTP non supporta l'autenticazione della *Proxy*. Questo tipo di autenticazione è necessaria per il GrAnT, perché i servizi che esso offre devono essere costruiti specificatamente in base alle richieste ed all'identità dell'utente.

Per risolvere tale problema il prototipo prevede un modulo *Strand* il quale aggiunge un campo *header* alla richiesta HTTP. Questo campo utilizza il classico metodo di identificazione basato sul nome dell'utente e sulla sua *password* al fine di identificare le risorse richieste. Il problema di tale approccio è che esso richiede l'avvio di un processo *Strand* su ogni macchina *Client* e ciò inevitabilmente aumenta il ritardo del protocollo di richiesta e risposta.

3.1.5 Conclusioni

In conclusione il sistema *Pan-Browser* consente la realizzazione di commenti indipendenti e condivisi con riferimento a documenti accessibili attraverso il *World Wide Web*. Tali commenti sono delle meta-informazioni salvate e gestite in un contesto diverso da quello dei documenti a cui si riferiscono; aspetto interessante è che i commenti possono essere riferiti all'intero documento od a sue porzioni interne.

Dal punto di vista organizzativo le annotazioni sono raggruppate in *Annotation Sets* gestiti da appositi *Annotation Servers*.

Per quanto riguarda l'implementazione sul *Browser* è realizzata un'interfaccia grafica che sfrutta le funzionalità delle maschere HTML, risultando così semplice, potente e standard. D'altro canto gli *Annotation Servers* hanno *performances* ridotte ed offrono solo le primitive funzionalità dei *Database* tradizionali, rendendo così l'intero sistema scarsamente flessibile soprattutto per quanto riguarda le modalità di ricerca delle annotazioni.

3.2 CoNote

CoNote è un esperimento con il quale le persone possono collaborare [D.LaLiberte 1995] quando lavorano contemporaneamente con un insieme di documenti condivisi [Prinz and Syri 1997]. L'aspetto più interessante di tale progetto consiste nel fatto che più persone possono lavorare con documenti condivisi ed effettuare commenti che risultano anch'essi condivisi tra i membri di un determinato gruppo. Le annotazioni possono riferirsi ad altre annotazioni; chi produce il documento può specificare chi lo può leggere e chi può leggere od aggiungere annotazioni[2]. Questo è quindi un meccanismo che consente, almeno in parte, il mantenimento della *Privacy* e della sicurezza dei dati. Infine CoNote è basato sul *World Wide Web* e quindi è possibile leggere le annotazioni con un normalissimo *Web Browser*.

3.2.1 Annotazioni

Le annotazioni possono essere considerate come delle note che si riferiscono a particolari sezioni di un documento oppure all'intero documento [Gramlich 1994]; a differenza di quanto si potrebbe fare con gli strumenti cartacei, le annotazioni in CoNote possono essere inserite solamente in quei punti del documento che l'autore ha predefinito durante la sua stesura. Associata ad ogni "punto di annotazione" di un documento CoNote ci può quindi essere una lista di annotazioni scritte in precedenza che possono apparire come illustrato qui sotto:

There are 2 annotations.
Un' annotazione di esempio Amanzio Rigamonti Mon,22 Sept 97 18:23:14 DTE
Una nota critica D.Socrates Fri,5 Oct 97 23:12:49 DTE

---

[2]Ciò significa che l'autore dell'annotazione può definire degli *Access Modifiers* per l'annotazione medesima.

L'annotazione di un documento CoNote è molto semplice e consiste essenzialmente nella definizione di ogni posto in cui l'annotazione potenzialmente può essere posizionata. Ci sono due modalità che possono essere adottate per la definizione dei posti in cui effettuare le annotazioni:

Notazione posizionale Con tale modalità occorre specificare l'insieme di *byte* del documento che costituiscono il testo che si vuole annotare. Ogni definizione di tal tipo deve essere effettuata nel *Document Definition File*. La modalità posizionale è un'ottima scelta nel momento in cui il documento è statico.

Notazione in linea Le definizioni in linea non sono effettuate tramite il *Definition File*, bensì tramite le marche HTML che si possono scrivere in linea nel documento che potrà essere annotato. Tale modalità è più conveniente rispetto a quella posizionale precedentemente descritta soprattutto se il documento viene modificato spesso.

Le annotazioni possono essere organizzate in conversazioni in cui un'annotazione può essere la risposta ad un'altra; in tal caso le annotazioni che non si riferiscono ad un documento bensì ad altre annotazioni sono appese sotto queste ed indentate verso l'interno. Se ci sono più risposte che si riferiscono ad una stessa annotazione esse sono riportate nell'ordine in cui sono state effettuate.

Questo è il presupposto alla creazione di conversazioni "molti a molti", in cui ci possono essere domande seguite da risposte e così via. In tal caso tutte le annotazioni che si riferiscono ad una stessa annotazione sono collegate tra loro e quindi appaiono in una lista che si trova al di sotto dell'annotazione di riferimento ed indentata rispetto a questa.

In tale prototipo vi è una maschera per effettuare una ricerca tra i documenti che sono già stati annotati e che visualizza una struttura ad albero relativa alle annotazioni che sono state effettuate in riferimento ad un determinato documento.

3.2.2 Utenti e ruoli

In CoNote un *group* è un insieme di persone che condividono una collezione di documenti; ogni persona nel gruppo ha un certo ruolo che può essere *viewer, reader, user* oppure *author*. L'ordine con cui sono elencati i ruoli è funzione crescente dei diritti sul documento che possiede la persona che riveste quel ruolo. Un *viewer* può visualizzare il documento annotato ma non può leggere le annotazioni ad esso associate, un *reader* può visualizzare le annotazioni ma non ne può aggiungere di nuove, un *user* può leggere ed aggiungere nuove annotazioni ed infine un *author* può leggere, aggiungere ed anche cancellare le annotazioni. È anche possibile che un utente non abbia alcun ruolo all'interno del *group* ed in questo caso egli non può nemmeno leggere i documenti; ugualmente una persona può assumere diversi ruoli nel *group*.

In tale prototipo le annotazioni sono salvate in un *Database* diverso da quello dei documenti; le annotazioni vengono associate al documento quando questo viene recapitato all'utente che l'ha richiesto[3]. Ciò è estremamente importante in quanto permette ad insiemi diversi di utenti di fare differenti annotazioni sullo stesso documento.

[3]Questa è la logica adottata anche nel prototipo proposto in tale testo.

Le annotazioni inoltre possono apparire nel documento solamente in locazioni predefinite anticipatamente dall'*author* e questo perchè HTML limita gli utenti dando loro la possibilità di selezionare solo determinati punti interni al documento.

Gli *authors* aggiungono documenti CoNote mediante la scrittura di un *Document Declaration File* nel quale vengono specificate le locazioni del documento ed i posti in cui è possibile appendere delle annotazioni. Inoltre gli *authors* mantengono un *roles file* in cui sono specificati i ruoli di tutti gli utenti.

In base a quanto sopra riportato appare evidente che il sistema necessita di un meccanismo di autentificazione per verificare l'identità di ogni utente; quando un utente richiede un documento il sistema visualizza una maschera di identificazione in cui occorre inserire il nome e la *password*. Se tutto va bene il documento è reso disponibile all'utente.

Visto che molti *Web Browsers* visualizzano il completo URL l'informazione di autentificazione potrebbe essere una chiave numerica casuale generata sopra il *login*.

L'*Annotation System* mantiene una lista di tutte le persone che sono autorizzate ad utilizzarlo. L'autentificazione è necessaria solo la prima volta che si accede all'*Annotation System*; tale autentificazione è implementata aggiungendo un campo chiamato *key* alla stringa che identifica ogni URL. Quando l'*Annotation System* consegna un documento esso appende automaticamente il campo *key* ad ogni *hyperlink* interno al documento. In tal modo quando si apre un *link* interno al documento esso possiede già la chiave d'accesso e quindi non c'è più bisogno di effettuare nuovamente l'autentificazione.

Per leggere le annotazioni occorre selezionarle ad esempio con un semplice click del *mouse* sopra di esse. Per spostarsi da un'annotazione all'altra si possono utilizzare dei comandi grafici che appaiono sullo schermo.

Ogni annotazione deve avere un soggetto, consistente in una piccola parte descrittiva, e del testo. Le annotazioni sono comunemente scritte in codice ASCII senza formattazione.

3.2.3 Annotazioni per documenti di gruppo

L'interfaccia concettuale tra CoNote ed un *author* è il *"Document Group"* ovvero una collezione di documenti con un certo nome. Ci possono essere diversi gruppi associati ad uno stesso sito ma ognuno, ovviamente, deve avere un nome diverso dagli altri.

Un *Document Group* consiste essenzialmente in un insieme di *Document Definitions*, una per ogni documento nel gruppo, ed una definizione dei ruoli dei vari componenti del gruppo. Tutti i documenti in un gruppo sono trattati allo stesso modo.

Per creare un nuovo *Document Group* occorre anzitutto scegliere un nome univoco, creare una *directory* unica all'interno del *files system* per quel determinato gruppo ed infine creare il *file* con la specifica dei vari ruoli. Tale *file* dei ruoli specifica quali utenti fanno parte del gruppo e cosa essi possono fare in funzione del ruolo assunto. Tale *file* ovviamente può essere cambiato dal gestore del gruppo.

I documenti annotati possono essere salvati ovunque nel *File System*, ma ovviamente il *Web Server* deve essere abilitato a leggerli; ciò significa che i *files* relativi a tali documenti devono risiedere in un *file system* a cui il *Web Server* può avere accesso.

3.2.4 Conclusioni

Il prototipo CoNote presenta alcune caratteristiche interessanti che val la pena riassumere facendo riferimento ai diversi criteri in base ai quali organizzare sistemi a supporto dell'annotazione digitale descritti nel capitolo 2.

In primo luogo il sistema CoNote offre la possibilità di effettuare annotazioni riferite all'intero documento od a porzioni di testo in esso contenute; inoltre gli utenti hanno la possibilità di effettuare annotazioni riferite ad altre annotazioni in base ad una logica ricorsiva.

Interessanti sono gli aspetti relativi alla definizione di specifici ruoli che possono essere assunti dagli utenti, ed i conseguenti meccanismi per la gestione della sicurezza dei dati tramite procedure di autentificazione basate sui diritti correlati a ciascun ruolo.

Nel complesso il sistema CoNote consente la cooperazione tra più utenti operanti in spazi condivisi, utilizzando strumenti standard che però, inevitabilmente, limitano la flessibilità dell'intero sistema.

3.3 ComMentor

ComMentor[4] è il nome che si riferisce ad un sistema prototipale sviluppato nell'ambito del *Stanford Integrated Digital Library Project*; tale prototipo è stato progettato al fine di costruire un'architettura in grado di fornire un supporto al meccanismo di condivisione delle annotazioni. Più precisamente esso definisce una piattaforma per mezzo della quale è possibile effettuare un processo di aggiunta di informazioni a documenti secondo varie modalità; tale architettura si posiziona al di sopra della già presente infrastruttura rappresentata dal *World Wide Web*.

Tale prototipo di ricerca è stato completato nel dicembre del 1994; la sua architettura consente l'arricchimento di documenti per mezzo di aggiunta di informazioni esterne e costituisce una possibile piattaforma tramite la quale gli utenti possono condividere, e quindi anche gestire, oggetti quali sono le annotazioni che si riferiscono a documenti che si possono trovare normalmente sul *World Wide Web*. Tale meccanismo è realizzato attraverso l'utilizzo di meta-informazioni che sono gestite in modo del tutto separato dal documento stesso e che risiedono su separati *Meta-Information Servers* dove vengono immagazzinate e gestite in termini di controlli d'accesso e diritti di utilizzo.

In tale logica è possibile raggiungere l'obiettivo di avere documenti personalizzati utilizzando *Databases* in cui sono memorizzati i documenti e meta-informazioni che consentono di costruire dinamicamente documenti in base alle specifiche ed ai desideri forniti dal particolare utente.

Di seguito verrà brevemente descritta, nei suoi punti generali, l'architettura di gestione delle meta-informazioni ed anche una sua implementazione, chiamata ComMentor, la quale è stata sviluppata per consentire l'annotazione delle pagine presenti sul *World Wide Web*.

Tale architettura è alla base dei concetti di fondo che hanno guidato lo sviluppo e la realizzazione del modello a supporto delle annotazioni descritto nei capitoli successivi di tale documento.

[4]Nome di un'implementazione prototipale sul *World Wide Web*; dal punto di vista del codice rappresenta un arricchimento del *Browser NCSA Mosaic* [developers 1993] e l'arricchimento di un *NCSA HTTP Server* con un insieme di *Server Scripts.*

3.3.1 Architettura

L'architettura di base del sistema presentato in tale sezione è riportata nella figura 3.3.

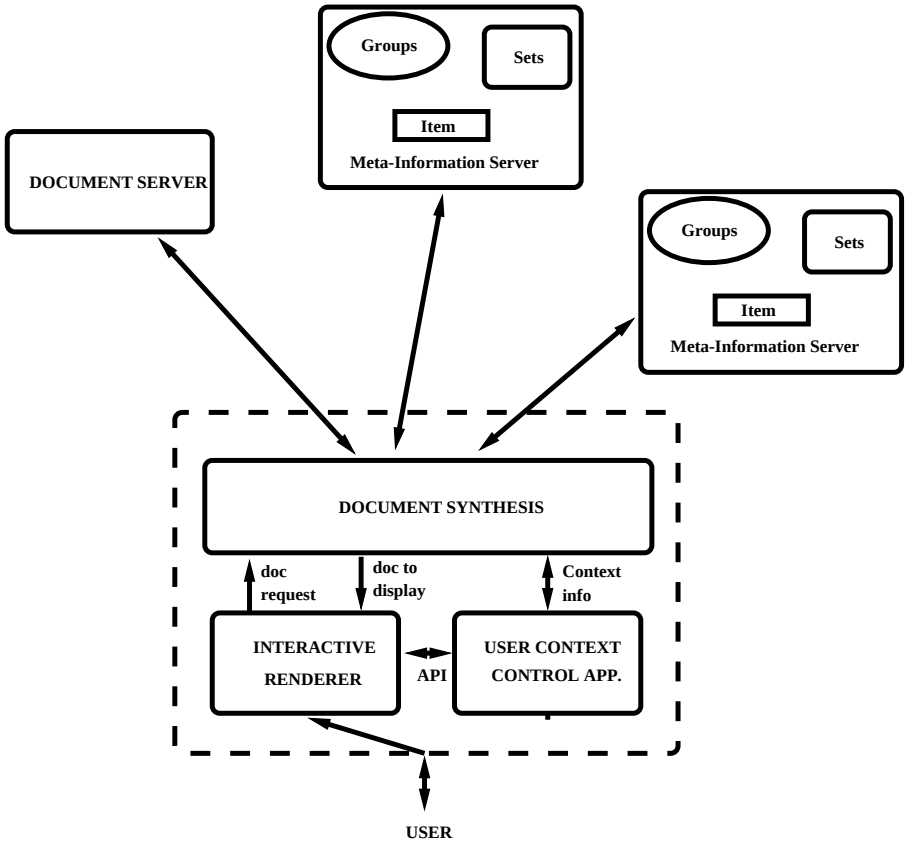

Figura 3.3: Architettura di base del prototipo ComMentor.

Come si può notare dalla figura 3.3 gli utenti interagiscono con un *Browser* al fine di recuperare i documenti che a loro interessano e che risiedono su vari *Document Servers*. Il *Browser* utilizzato non è un normale *Web Browser* bensì è progettato specificatamente per lo svolgimento di particolari attività tra cui anche quella di sintesi tra le annotazioni ed i documenti a cui esse si riferiscono.

In aggiunta a ciò vi sono i *Meta-Information Servers* dai quali possono essere recuperate molte informazioni utili. Le modalità di reperimento di tali informazioni sono presentate di seguito nella descrizione del protocollo di funzionamento. Le meta-informazioni (come le annotazioni) sono organizzate in insiemi ai quali hanno accesso i membri che appartengono a determinati gruppi.

Come si osserva dalla figura 3.3 il *Browser* è composto da un *Interactive Render* quale può essere una normale applicazione per consentire all'utente di interagire con il sistema, da un *Document Synthesis Module* il quale si occupa di interrogare i *Servers* al fine di recuperare documenti e/o meta-informazioni ed infine da un *User Context Control Application* che è un modulo

che comunica con gli altri due allo scopo di gestire informazioni relative a funzionalità estese nell'interazione del sistema con l'utente.

3.3.2 Organizzazione degli utenti e delle annotazioni

Di seguito verrà descritta la struttura interna (figura 3.4) di un *Meta-Information Server* che è, a mio avviso, il componente più importante dell'intera architettura. La figura 3.4 mostra le entità di base della struttura che sono: *Members*, *Groups* ed *Annotation Sets*.

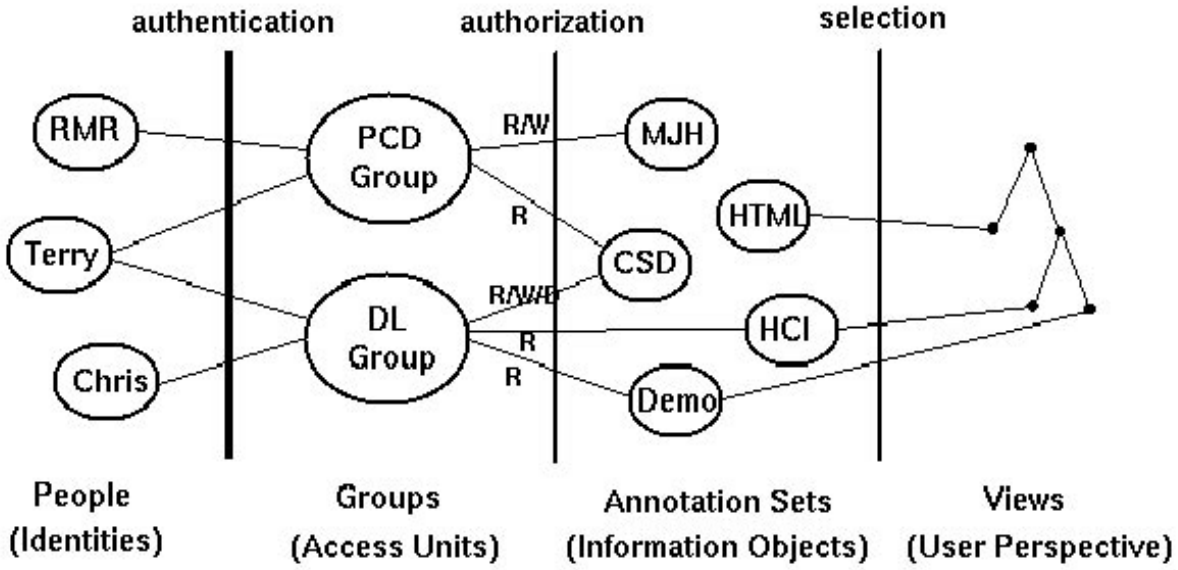

Figura 3.4: Struttura interna del *Meta-Information Server.*

L'architettura è basata sugli *Annotation Sets*; ogni annotazione deve appartenere ad un particolare insieme di annotazioni ed annotare un documento eventualmente in una specifica locazione.

Ogni insieme di annotazioni è associato con un particolare *Annotation Server* che è identificabile tramite un preciso indirizzo (come l'URL). L'*Annotation Set* tiene l'indice di tutte le annotazioni che appartengono all'insieme ma non necessariamente mantiene il loro contenuto. Tale *Annotation Server* inoltre è in generale distinto dal *Server* su cui risiede il documento a cui l'annotazione si riferisce.

Da quanto detto si può quindi pensare l'insieme di annotazioni come un unico documento distribuito costituito da annotazioni che sono associate con pagine HTML che possono risiedere ovunque sulla rete.

Il controllo degli accessi è eseguito sull'insieme di annotazioni; ci possono essere anche diversi *Annotation Sets* associati con un gruppo, ognuno dei quali possiede obiettivi diversi. La cosa importante da osservare è che il meccanismo di autentificazione degli accesi è una questione che viene mantenuta ortogonale al progetto; ogni meccanismo di autentificazione può essere utilizzato. In tale struttura tutte le meta-informazioni possono essere pensate come oggetti.

Il *Browser* fornisce semplici meccanismi per identificare un posto nella pagina corrente e per scrivere commenti interni ad un *set* specificato dall'utente. Per aggiungere un'annotazione l'utente semplicemente seleziona una regione nel documento che deve essere annotato ed il *Browser* automaticamente identifica eventuali ridondanze del testo selezionato all'interno del documento.

L'utente può inoltre aggiungere la sua annotazione ad un insieme di annotazioni già esistenti. Le annotazioni possono essere indicate in un'interfaccia in svariati modi; il *Browser* sviluppato in tale prototipo utilizza il meccanismo delle marche situate nella posizione in cui si vuole aggiungere un'annotazione; inoltre anche le immagini possono essere annotate e porzioni di testo possono essere evidenziate con una barra luminosa per indicare l'annotazione di quel pezzo di testo. Inoltre l'interfaccia del *Browser* include meccanismi che supportano diversi stili che possono essere adottati per la lettura delle annotazioni.

Le annotazioni possono essere filtrate in base a diversi criteri; un tipico modo è quello di filtrare le annotazioni in base ad una certa etichetta in modo tale da visualizzare solo documenti che stanno in un certo insieme. Dal momento che le annotazioni sono esse stesse dei documenti, esse possono essere ricorsivamente annotate. Il sistema inoltre consente di mantenere una lista di riferimenti a tutte quelle annotazioni che sono state referenziate più di recente.

Altro concetto molto importante è quello di meta-informazioni, che sono quelle informazioni utilizzate per trasmettere informazioni relative ad un documento in modo uniforme e meccanico. Tale prototipo utilizza, al fine di descrivere le meta-informazioni, un linguaggio noto come PRDM (*Partial Redundant Descreptive Meta-language*). Esso è un linguaggio ad oggetti che è stato sviluppato nell'ambito dello *Stanford Integrated Digital Library Project*.

Nel prototipo altro componente molto importante è la *Merge Library* che è un insieme di procedure che sintetizzano il documento finale ottenuto fondendo il documento base con le annotazioni che ad esso si riferiscono. Tali procedure sostanzialmente prendono il documento selezionato e la lista PRDM dei commenti che ad esso si riferiscono, producendo un documento dove i commenti sono in linea con il documento originale. Tali procedure ovviamente sono specifiche a seconda del tipo del contenuto del documento.

Il metodo utilizzato per attaccarsi ad un testo HTML od ad un normale testo è basato sulle strutture ad albero che descrivono la posizione delle stringhe nel documento [Knuth 1973]. Ogni commento, che è un oggetto, ha associato una porzione di testo sottolineato ed un identificatore che definisce la posizione del testo nel documento. Tale metodo è molto efficiente in quanto gli identificatori della posizione di una stringa nel testo consentono di lavorare ad un livello di granularità molto fine ed inoltre sono un metodo molto robusto in quanto seguono le modifiche del documento che sta alla base.

3.3.3 Modalità di interazione tra *Client* e *Server*

Le modalità di interazione tra *Client* e *Server* rappresentano un aspetto di fondamentale importanza nell'ambito del prototipo ComMentor[5]. In tale interazione le operazioni più rilevanti sono quelle relative al recupero di un documento e delle annotazioni ad esso riferite, il collegamento ad un gruppo e l'aggiunta di un *Annotation Set*.

Per quanto riguarda il recupero del documento e delle relative annotazioni , esso avviene accedendo rispettivamente al *Document Server* ed agli *Annotation Servers*.

Il primo passo riguarda il recupero del documento desiderato (figura 3.5).

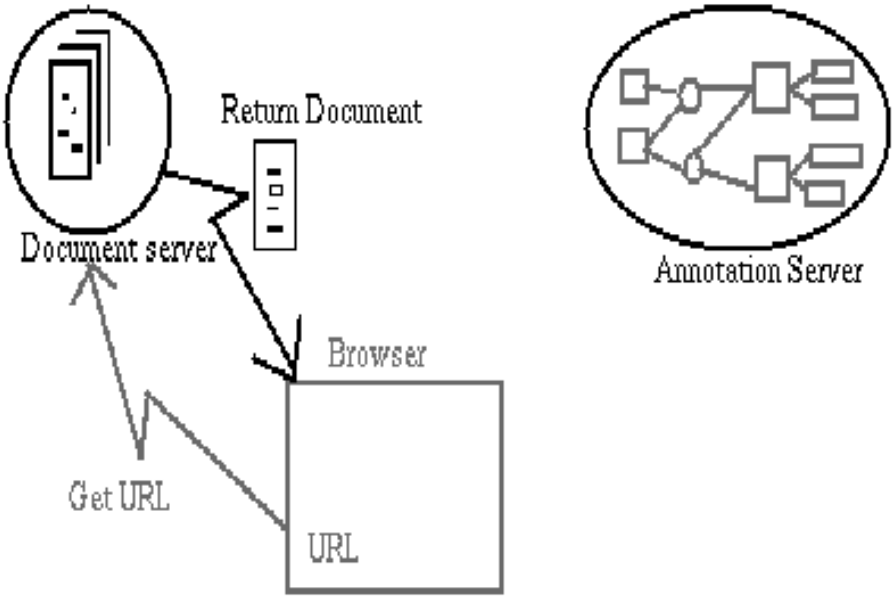

Figura 3.5: Recupero del documento.

Tale operazione è la classsica porcedura implementata da un *Browser standard* che consente di navigare sulla rete.

Il secondo passo consiste nel richiedere le annotazioni che si riferiscono ad un certo documento (figura 3.6).

Come si può vedere nella figura 3.6 il *Browser* invia all'*Annotation Server* una richiesta per ottenere le annotazioni relative ad un certo documento identificato con il suo URL.

Il *Browser* deve inoltre specificare all'*Annotation Server* in quale *Annotation Set* deve andare a reperire le annotazioni da attivare. A questo punto l'*Annotation Server* restituisce il proprio responso al *Browser* (figura 3.7).

[5]Le regole di comunicazione tra *Client* e *Server* rappresentano un aspetto estremamente delicato in fase di progettazione di un qualsiasi sistema per l'annotazione.

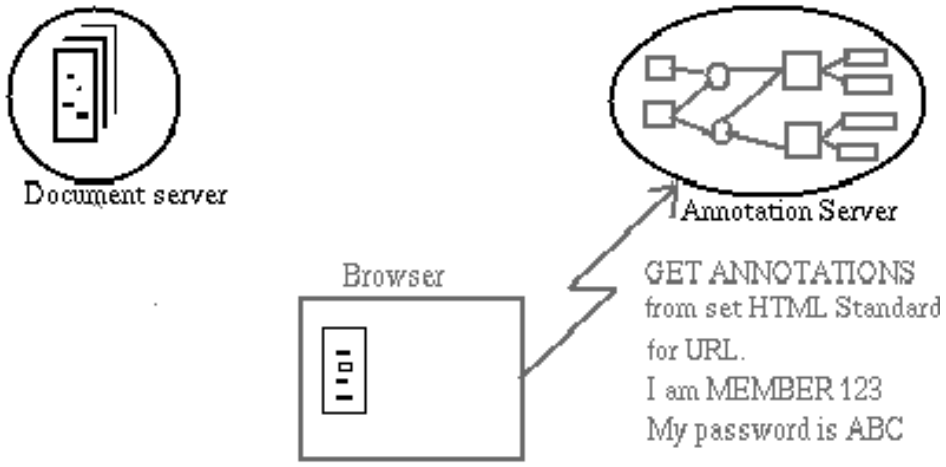

Figura 3.6: Richiesta delle annotazioni relative ad un documento.

Come si può notare dalla figura 3.7 l'*Annotation Server* restituisce una serie di meta-informazioni che il *Client* fonderà in maniera opportuna con il documento di base.

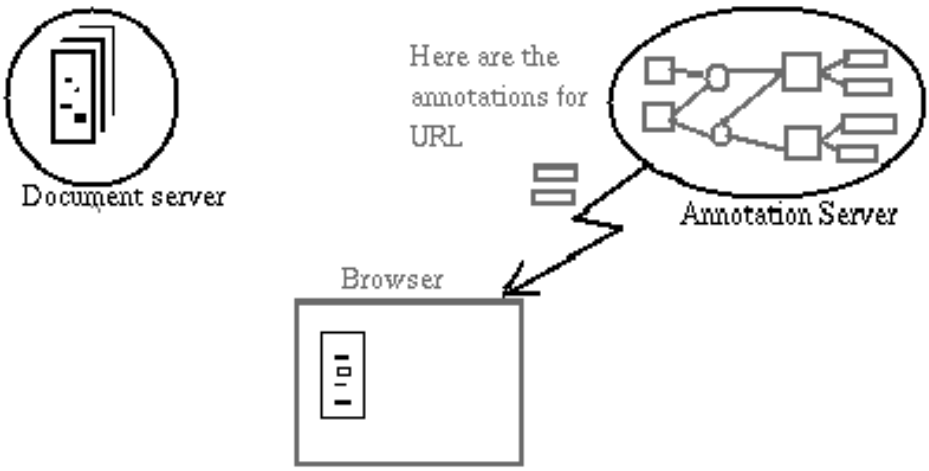

Figura 3.7: Restituzione delle annotazioni al *Browser*.

A questo punto sul *Client* avviene il passo più delicato dell'intera procedura ovvero il *Merge* tra il documento e le annotazioni (figura 3.8). Tale fusione viene effettuata da una serie di procedure che utilizzano la struttura ad albero che descrive il documento e la struttura che identifica le informazioni ridondanti interne al documento; in tal modo esse possono identificare la corretta posizione interna al documento a cui la particolare annotazione si riferisce. Tali funzioni sono progettate in modo tale da essere robuste ai cambiamenti del documento di base a cui riferire le annotazioni.

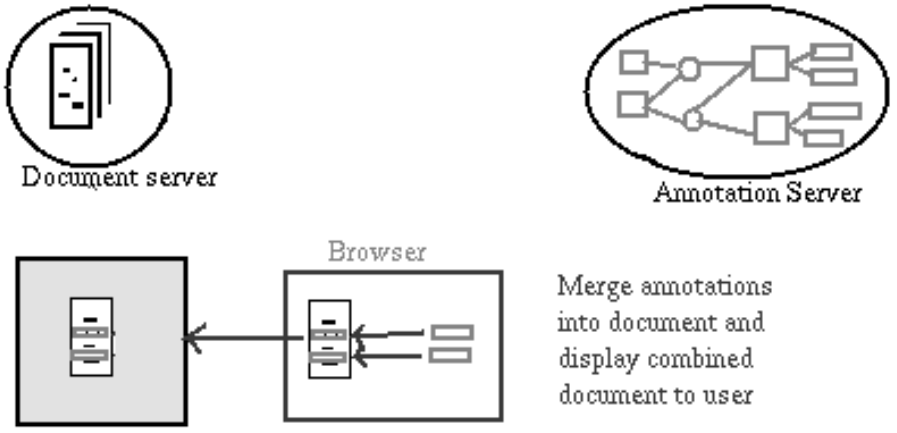

Figura 3.8: *Merge* delle annotazioni con il documento di riferimento.

Altro aspetto relativo all'interazione tra *Client* e *Server* è rappresentato dal collegamento ovvero dall'entrata di un utente in un gruppo[6]. Infatti prima di leggere i commenti appartenenti ad un determinato insieme, l'utente deve appartenere ad un gruppo che ha l'accesso all'*Annotation Set* a cui appartiene l'annotazione richiesta.

3.3.4 Conclusioni

In conclusione il prototipo ComMentor presenta sostanzialmente un'architettura *Client/Server* che consente condivisione in un ambiente distribuito.

Attraverso il sistema ComMentor è possibile annotare un qualsiasi documento presente sul WWW; le annotazioni vengono gestite separatamente dai documenti a cui si riferiscono. Le annotazioni sono organizzate in *Annotation Sets* residenti su opportuni *Meta-Information Servers*.

Il sistema è interamente progettato in modo tale che il meccanismo di autentificazione degli utenti sia ortogonale alle scelte progettuali effettuate.

Nel complesso l'approccio adottato in fase di implementazione e le tecnologie utilizzate non consentono l'ottenimento di soluzioni *standard* ai vari problemi; ciò però permette un elevato grado di flessibilità nello svolgimento delle diverse operazioni di gestione delle annotazioni.

3.4 NCSA Project

NCSA è un sistema sviluppato per supportare annotazioni pubbliche e di gruppo [Braverman 1996]. La realizzazione di tale prototipo è stata effettuata prestando un'attenzione particolare alla costruzione di un sistema dotato della proprietà di scalabilità.

Una soluzione ad un problema si può definire scalabile se essa continua ad essere valida, e soprattutto efficente, anche se molte variabili del problema cambiano assumendo dei valori e

[6]Per tale funzionalità non è riportata una spiegazione dettagliata. Per il lettore interessato si consiglia di fare riferimento a [Roscheisen *et al.* 1997].

riferendosi a contesti di più ampia scala. Alla luce di ciò non è difficile capire come una soluzione che risulta essere ottimale per un piccolo problema potrebbe divenire impraticabile o totalmente inefficiente nel momento in cui lo stesso problema si riferisce ad un contesto più ampio.

Per comprendere a fondo il problema della scalabilità per le annotazioni occorre capire in primo luogo il problema della scalabilità riferito al *Web* stesso. Alcune delle limitate risorse del *World Wide Web* sono la larghezza di banda della rete e la capacità computazionale di *Servers* e di *Clients*. Un altro limite della risorse di rete è la ridotta capacità umana di produrre e consumare le risorse.

La larghezza di banda della rete e il potere computazionale sono generalmente incrementabili, ma non così rapidamente da poter mantenere la crescita esponenziale del numero di utenti della rete.

Una delle sfide più significative del *Web* è quindi il numero di utenti il quale, tra non molto, sarà dell'ordine del miliardo.

Le annotazioni di documenti hanno almeno tanti problemi di scalabilità quanti ne hanno i documenti stessi[7]; oltre a tali problemi vi sono anche numerose altre variabili che influenzano il problema della scalabilità delle annotazioni. Tra queste le più rilevanti sono il numero di lettori di un documento che sono anche interessati alla lettura delle annotazioni associate al documento, il numero di lettori che desiderano essere notificati circa i cambiamenti o le nuove annotazioni relative a determinati documenti, la quantità di informazioni che deve essere mantenuta in riferimento a tali lettori ed al contenuto che essi leggono, la frequenza con cui i lettori chiedono di essere informati circa i cambiamenti di loro interesse, il numero di annotazioni che possono essere associate ad ogni documento, il numero di documenti che possono essere supportati da un *Annotation Server*, la durata della vita delle annotazioni ed infine le modalità in base alle quali i *Clients* vengono a conoscenza delle annotazioni attive associate ad un certo documento.

Il problema principale per le annotazioni pubbliche consiste nel fatto che il pubblico è essenzialmente un largo gruppo che include chiunque.

3.4.1 Il modello

Nel modello proposto le annotazioni di gruppo sono distribuite tra tutti i *Servers* che supportano i gruppi di annotazioni. Ogni *Server* specifica quale insieme di documenti da annotare esso è in grado di servire e quale insieme di utenti hanno il permesso di creare e di leggere le annotazioni di tale *Server*.

In una struttura come questa un singolo documento potrebbe avere annotazioni anche su alcuni, diversi *Annotation Servers*. Al fine di distribuire il carico, un singolo gruppo di utenti potrebbe essere associato con alcuni *Group Annotation Servers* ognuno dei quali gestisce annotazioni per un differente insieme di documenti. I *Clients* potrebbero essere abilitati a sapere quali insiemi di documenti sono associati ad ogni gruppo (ad esempio attraverso la specifica del loro URL) in modo tale da evitare inutili richieste di annotazioni a *Servers* che non le hanno. Questo comportamento cooperativo è pensato per incrementare la scalabilità del gruppo di annotazioni.

[7] Questo perché teoricamente ogni documento potrebbe essere annotato con una o più annotazioni.

In tale contesto viene definito un protocollo di alto livello che sta al di sopra dell'*HTTP protocol* con lo scopo di consentire la comunicazione tra *Clients* e *Servers* per quanto riguarda la disponibilità di annotazioni, i meta-dati relativi a ciascuna annotazione ed il testo delle annotazioni stesse. Il protocollo inoltre supporta anche la creazione, l'*editing* e la distruzione delle annotazioni. In tale modello sono supportate solo annotazioni accessibili tramite il protocollo HTTP che si riferiscono a documenti HTTP.

Il protocollo è molto simile per entrambi i tipi di annotazioni, di gruppo oppure pubbliche; infatti un'implementazione di un *Group Annotation Server* potrebbe essere utilizzata anche per realizzare un *Public Annotation Server*.

La figura 3.9 illustra come un *Client* potrebbe ottenere un documento e le annotazioni pubbliche o di gruppo ad esso associate.

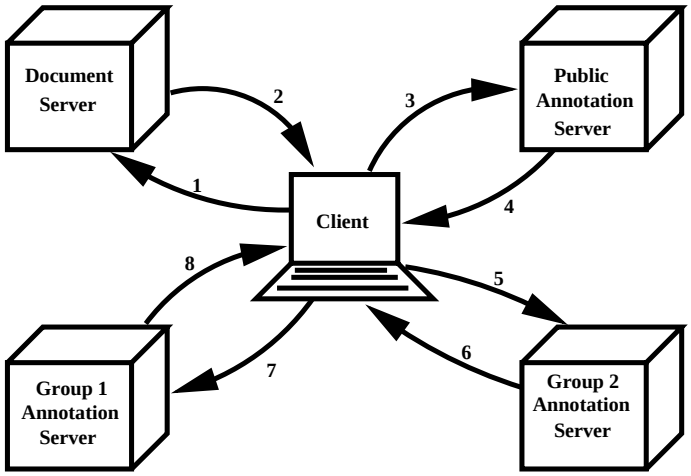

Figura 3.9: Protocollo per il reperimento di annotazioni pubbliche e di gruppo.

È riportata di seguito una breve descrizione del protocollo il quale è di estrema importanza per la comprensione del meccanismo di annotazione adottato in tale progetto.

1. Il *Client WWW* richiede un documento da un *HTTP Server*. Questa fase avviene con la solita procedura di reperimento di un documento sulla rete;

2. Il *Server HTTP* ritorna il documento con davanti un *Annotations-CGI header* che riporta l'URL del *Server* che fornisce i servizi per la gestione delle annotazioni pubbliche;

3. Il *Client* interroga tale *Server* chiedendo la lista delle annotazioni associate con l'URL che si riferisce al documento di base;

4. Il *Server* delle annotazioni pubbliche ritorna la lista nel formato richiesto dal *Client*;

5. Il *Client* interroga un *Server* che fornisce i servizi per le annotazioni di gruppo. Ciò avviene nello stesso modo visto al punto 3.

6. Il *Server* che gestisce le annotazioni di gruppo ritorna una lista delle annotazioni associate al documento base. Tale lista ovviamente è accessibile solo dai membri del gruppo.

7. Il *Client* interroga un secondo *Server* che supporta annotazioni relative ad un diverso gruppo.

8. Tale secondo *Server* ritorna una nuova lista di annotazioni del gruppo da lui gestito.

Nel momento in cui un *Client* richiede un documento ad un *Server*, il *Server* risponde con il documento ed una linea di intestazione che referenzia uno o più *Servers* che gestiscono le annotazioni pubbliche che si riferiscono a quel documento specifico.

Per quanto riguarda le annotazioni di gruppo il meccanismo è molto simile; quando un utente accede ad un *WWW Client*, una configurazione di parametri indica di quale gruppo di utenti è membro l'utente stesso e dove si trova il *Server* che gestisce le annotazioni per ogni gruppo. Il *Client* quindi si autentifica con ogni *Server* di gruppo che a sua volta ritorna una descrizione (ad es. attraverso l'URL) dell'insieme dei documenti per i quali esso fornisce le annotazioni.

La parte restante del protocollo è la stessa sia per le annotazioni pubbliche che per le annotazioni di gruppo; il *Client* richiede ad ogni *Annotation Server* di ritornare informazioni relative a tutte le annotazioni relative al documento in questione.

L'*Annotation Server* ritorna questa informazione usando lo specifico formato richiesto dal *Client*, come ad esempio il formato HTML con *hyperlinks* per ogni annotazione.

Il protocollo per la creazione delle annotazioni è molto semplice; il *Client* notifica all'*Annotation Server* che l'utente vuole aggiungere un'annotazione per un particolare documento e quindi l'*Annotation Server* controlla se l'utente è abilitato a fare ciò. Il *Client* può allora spedire l'annotazione che deve quindi essere preparata sul *Client* stesso.

L'*editing* è molto simile alla creazione con la differenza, ovviamente, che ora l'annotazione esiste già e deve essere inviata indietro al *Client*.

Le annotazioni possono essere cancellate dal loro *author* oppure dal proprietario del documento a cui si riferiscono. L'*Annotation Server* prima effettua l'autentificazione di colui che vuole cancellare l'annotazione e poi, se egli ha i diritti richiesti, esegue la cancellazione.

3.4.2 Notificazione e *Polling*

In un meccanismo come quello descritto il numero di annotazioni che si riferiscono ad un documento può crescere enormemente con il trascorrere del tempo; l'utente quindi non vorrà rivedere, ogni volta che visita il documento, tutte le annotazioni ma soltanto quelle nuove di cui non è ancora a conoscenza. Inoltre l'utente dovrebbe avere la possibilità di sapere quando vi sono nuove annotazioni relative ad un certo documento in modo tale da non dover rivisitare in continuazione il documento con una notevole perdita di tempo.

A tal proposito gli approcci che possono essere seguiti per risolvere il problema sono due: *Polling* e *Notification*. In tale logica vi sono *tradeoffs* che inevitabilmente si devono accettare nella scelta dell'uno piuttosto che dell'altro approccio. Il progetto NCSA fornisce entrambi questi meccanismi che possono essere scelti da ogni utente che desidera avere informazioni circa l'esistenza di nuove annotazioni.

Polling: tale meccanismo consente di sapere cosa c'è di nuovo, nel nostro caso riguardo le annotazioni, attraverso l'implementazione di un particolare filtro che fornisce la data dell'ultima

richiesta effettuata dall'utente. In tale logica il *Client* potrebbe comunicare al *Server* la data dell'ultima richiesta oppure il *Server* si dovrà ricordare la data dell'ultima richiesta di ogni utente sottoscritto. Per sapere periodicamente cosa c'è di nuovo in termini di annotazioni relative a determinati documenti si potrebbe implementare un programma mandato in esecuzione sul lato *Client*, il quale automaticamente sonda i *Servers* di questi documenti per trovare cosa c'è di nuovo creando un unico documento HTML contenente tutte queste informazioni.

Questo meccanismo potrebbe funzionare bene per quei documenti che sono annotati frequentemente[8], ma per quei documenti non annotati frequentemente tale meccanismo comporterebbe un carico inutile per i *Servers* stessi.

Notification: per documenti che non cambiano frequentemente, oppure per gli utenti che vogliono essere ricordati solo saltuariamente dei cambiamenti avvenuti, è più efficiente che sia il *Server* ad informare gli utenti dei cambiamenti verificatisi attraverso un meccanismo di notificazione al *Client*. Tale soluzione evita i problemi relativi ad un continuo *Polling* del *Client*, che è un meccanismo sempre attivo anche se nulla è cambiato. D'altro canto la notificazione ha un costo proporzionale al numero di utenti che devono essere messi a conoscenza del cambiamento.

Per supportare tale meccanismo il *Server* potrebbe mantenere una lista principale per ogni documento. Per ogni cambiamento per il quale è richiesta la notifica potrebbe essere inviata una *e-mail* ad ogni membro di tale lista principale. Questo meccanismo risulta essere efficiente solo se, come spiegato sopra, il numero degli elementi della lista non è troppo elevato.

3.4.3 Conclusioni

Il prototipo NCSA è stato sviluppato ponendo particolare attenzione alla realizzazione di soluzioni dotate di elevata scalabilità. In tale logica si inseriscono soprattutto i meccanismi di notificazione attraverso i quali gli utenti vengono portati a conoscenza di eventuali nuove annotazioni realizzate nell'ambito del sistema a loro supporto.

Le annotazioni vengono organizzate in *Annotation Sets* e gestite su appositi *Servers*; il sistema consente inoltre l'annotazione di un qualsiasi documento presente sul *World Wide Web*.

La ricerca di scalabilità e di flessibilità inevitabilmente è a discapito di soluzioni *standard* e portabili.

3.5 Multivalent Annotation

Questo è uno dei modelli più interessanti che sta alla base di molti dei principi fondamentali riguardanti il problema dell'annotazione e riportati, attraverso una sistematica classificazione,

[8] In tal caso un meccanismo di *Polling* offre la possibilità di conoscere rapidamente tutte le nuove annotazioni effettuate.

nel capitolo 2. La proposta formalizzata in tale modello si basa sul presupposto che, per una corretta annotazione digitale, le annotazioni devono presentare una serie di proprietà che sono di seguito riportate brevemente.

Apparire in loco ciò significa che le annotazioni devono apparire nel documento stesso;

Elevata espressività ciò significa che le annotazioni devono essere disponibili a diversi livelli di granularità;

Indipendenza dal formato ciò significa che il sistema delle annotazioni deve poter lavorare con documenti digitali di vari formati;

Estendibilità e componibilità ciò significa che individui, gruppi, lettori e terze parti devono essere in grado di sviluppare un loro proprio stile di annotazione, il quale potrebbe essere il più vario possibile;

Distribuite ed aperte l'annotazione deve essere un processo completamente aperto ed il documento annotato deve essere un oggetto esistente come un servizio di rete distribuito;

Indipendenti dalla piattaforma di sviluppo essendo il documento annotato un oggetto presente sulla rete, le annotazioni così come i documenti a cui esse si riferiscono devono poter essere visualizzate su di una qualsiasi piattaforma;

Robustezza i documenti e le annotazioni possono ovviamente essere cambiati in modo asincrono; in tal senso le annotazioni devono essere robuste al fine di sopravvivere ad ogni minimo cambiamento del documento a cui si riferiscono;

La soluzione proposta chiamata *Multivalent Annotations*, cerca di soddisfare la maggior parte dei requisiti sopra riportati. Tale modello di annotazione si riferisce ad un preciso modello per i documenti noto come *Multivalent Document Model* [Phelps and Wilensky 1996].

3.5.1 *Multivalent Documents*

Tale modello noto come *Multivalent Document Model(MVD)* [Phelps and Wilensky 1996], considera un documento come composto da una serie di oggetti che rappresentano il suo contenuto (es. testo, immagini, video, *hyperlinks*); tali oggetti vengono denominati *layers*. Il documento inoltre è costituito anche da programmi dinamicamente precaricati noti come *behaviors*.

Ad esempio un *behavior* "ricerca" implementerà un tradizionale metodo di ricerca di una stringa all'interno del documento a cui si riferisce.

L'architettura MVD è stata implementata in Java, ed è stata applicata alle oltre 200.000 immagini della collezione presente nella biblioteca digitale di Berkeley.

Sostanzialmente in tale sistema quindi un documento è un insieme di *layers* e di *behaviors* concettualmente legati tra loro; per ricaricare il documento completo le *frameworks* a disposizione vanno a prendere questi pezzi sopra la rete e li compongono assieme.

3.5.2 *Multivalent Annotations*

Una serie di prove sperimentali ha dimostrato che gli utenti vogliono considerare le annotazioni come uno strato separato dal documento e percettibilmente distinto dal testo sottostante[9].

Il MDV descritto nella sezione precedente è esso stesso naturalmente predisposto alla logica delle annotazioni in cui gruppi di annotazioni, fatte da un singolo autore, possono essere collezionate assieme come un singolo *layer* il quale a sua volta può, come previsto dall'aspetto centrale dell'organizzazione del sistema, essere composto con un documento di base o con altre annotazioni [O'Hara 1997]. Le *Multivalent Annotations* quindi sono implementate da, o come, specializzati *behaviors*.

Il modello comprende tre diverse tipologie di annotazioni che fanno rispettivamente uso di (1) intervalli di elementi (spesso caratteri), (2) regioni geometriche oppure (3) strutture interne all'albero del documento.

Span Annotations

Le *Span Annotations* sono annotazioni che fanno uso di intervalli, ovvero di oggetti che si estendono in modo continuo da un punto all'altro del documento. Tali intervalli sono quindi implementati come oggetti attaccati al loro punto di partenza e di fine all'interno del documento. Tali oggetti possono ricevere eventi generati da azioni dell'utente come ad esempio i clicks del *mouse*.

Possono anche essere definiti dei *behaviors* che si occupano specificatamente di gestire tali oggetti, creandoli e distruggendoli, salvandoli e recuperandoli.

Forse l'esempio più semplice di *Span Annotations* consiste in una versione digitale della sottolineatura di frasi in un documento; la sottolineatura è infatti un intervallo che si estende da un punto di inizio ad un punto di fine interni al documento.

Geometric Region Annotations: Lenses

Nella sezione precedente è stato spiegato come le *Span Annotations* consentano di associare annotazioni a specifici punti del testo e ciò significa che esse sono in grado di lavorare ad un livello di granularità molto fine.

Un altro tipo di *Multivalent Annotations* sono le cosiddette *Lenses*[10] le quali fanno riferimento a regioni geometriche dell'intero documento. Anche in tal caso ci possono essere diversi tipi di *Lenses* che, a loro volta, possono essere combinate assieme agli effetti di altri *behaviors*. Le *Lenses* si possono anche sovrapporre ed in tal caso gli effetti si compongono.

Anche le *lenses* possono ricevere eventi con i quali esse possono chiudersi, essere spostate oppure essere trasformate.

[9] Questo è un principio che ricorre in numerosi sistemi e che è stato adottato anche nella progettazione del sistema proposto.

[10] Questa modalità di annotazione consiste essenzialmente nel mettere in evidenza una porzione di testo attraverso uno *zoom* della stessa.

Structural Annotations

Le *Structural Annotations* sono annotazioni che si agganciano all'albero che definisce la struttura del documento di riferimento. I documenti infatti sono generalmente strutturati gerarchicamente in quanto un libro, ad esempio, contiene capitoli, che a loro volta contengono sezioni, che a loro volta contengono sottosezioni e così via.

Le *Structural Annotations* fanno riferimento quindi a specifiche parti della struttura che costituisce il documento.

Combinazione delle annotazioni

Tutti e tre i tipi di annotazioni descritti nelle sezioni precedenti possono essere combinati assieme in uno stesso documento. In tal caso ci potrà essere una combinazione degli effetti prodotti da ogni tipo di annotazione.

Ad esempio l'effetto di una *Span Annotation* potrebbe combinarsi con quello di una o più *Lenses* ed inoltre a quello stesso paragrafo potrebbe essere associata una *Structural Annotation*[11].

Una logica di lavoro di tal tipo risulta essere di estrema utilità e soprattutto consente di soddisfare appieno le esigenze degli utenti che possono essere le più varie possibili a seconda del campo di competenza e degli interessi che ognuno possiede, i quali sono inevitabilmente, ma anche fortunatamente, diversi.

3.5.3 Conclusioni

Il modello *Multivalent Annotations* offre un potente meccanismo di annotazione attraverso il quale l'utente ha la possibilità di effettuare annotazioni di vario tipo in base a criteri molto elastici e componibili; ciò va a vantaggio dell'elevata flessibilità dell'intero sistema.

Lo svantaggio consiste nel fatto che tale approccio non risulta essere *standard* ovvero dipende fortemente dalla specifica implementazione del sistema, limitandone la possibile diffusione su larga scala.

Altra caratteristica importante del sistema *Multivalent Annotations* è che le annotazioni ed i documenti sono gestiti separatamente in modo tale da poter ottenere soluzioni scalabili ai vari problemi che sono strettamente correlati alla procedura di annotazione.

3.6 AKI

AKI è un sistema di annotazione a supporto dell'uso cooperativo di informazioni presenti sul *World Wide Web*; tale sistema consente il recupero, l'amministrazione, la diffusione, la rappresentazione e la notificazione di annotazioni che si riferiscono ad oggetti referenziati da un preciso URL e che risiedono sul *World Wide Web*. Tale sistema è stato sviluppato sulla base di una definizione che per annotazione considera tutto ciò che costituisce un completamento del contenuto

[11] La realizzazione di un meccanismo di tal tipo è estremamente complessa e potrebbe rappresentare uno dei possibili sviluppi futuri del sistema implementato.

di un documento tramite osservazioni e commenti personali. Le annotazioni possono essere analizzate sotto diverse prospettive ed i criteri che stanno alla loro base sono riportati nel capitolo 2.

3.6.1 Caratteristiche generali

Scopo principale del sistema AKI è quello di riuscire a raggiungere il consenso da parte della maggior parte degli utenti del *World Wide Web*. Per tal motivo il sistema è stato realizzato secondo un principio di indipendenza dal particolare *WWW-Client* o *WWW-Server*. Il sistema si basa essenzialmente sul protocollo HTTP con il quale l'utente può lavorare tranquillamente utilizzando il proprio *WWW-Browser*.

In tale sistema gli utenti sono organizzati in gruppi che vengono gestiti da un amministratore del gruppo[12]; ogni gruppo è costruito per soddisfare determinati interessi. Per tal motivo gli utenti possono appartenere anche a più gruppi che vengono strutturati gerarchicamente. I gruppi inoltre possono essere "aperti", a cui ogni utente può appartenere, oppure "privati" che sono quei gruppi a cui possono appartenere solo alcuni utenti.

3.6.2 Architettura del sistema

Il sistema AKI è costituito da tre elementi principali consistenti in un *Server* che gestisce le annotazioni, una base di dati ed un particolare *Proxy Server* (vedi figura 3.10).

Come si può notare dalla figura 3.10 nella base di dati vengono memorizzati i dati degli utenti, le informazioni relative agli interessi dei gruppi e le annotazioni.

La base di dati è amministrata dall'*Annotation Server* che può essere visto anche come *Gateway* tra il *Browser* e la base di dati stessa.

Il *CGI-Konnector* consiste in una quantità di *CGI-Scripts* che si occupano di istanziare le annotazioni e di fornire notizie a loro riguardo.

Il modulo *Email-Konnector* serve a realizzare il meccanismo di notificazione *off-line*, mentre l'*Applet-Konnector* si occupa della notificazione *online*.

La notificazione automatica attraverso la visita di un URL già annotato viene realizzata con l'aiuto del *Proxy-Server*; per tale ragione il *Proxy-Server* deve essere registrato, prima dell'uso, nel *WWW-Browser*. Il *Proxy-Server* svolge il compito di controllare il flusso HTTP[13] e notificare al modulo *Proxy-Konnector* dell'*Annotation Server* gli URLs che sono già stati visitati.

L'*Annotation Server* guarda quindi nella base di dati e verifica se vi sono annotazioni che si riferiscono agli URLs che gli sono stati inviati dal *Proxy-Server*; nel caso in cui la ricerca da esito positivo l'*Annotation Server* informa l'utente con l'aiuto dei *CGI-Konnector, Email-Konnector* ed *Applet-Konnector*.

[12] Il concetto di gruppo nel sistema AKI è analogo al concetto di Progetto che è analizzato nel capitolo 5. Anche i progetti hanno un *Project Administrator* ed un insieme di elementi che costituiscono il gruppo di progetto. Inoltre i progetti, come i gruppi nel sistema AKI, possono essere organizzati gerarchicamente.

[13] In tale controllo è compreso anche l'insieme di operazioni necessarie per effettuare il *merge* tra documento annotato ed annotazioni che ad esso si riferiscono.

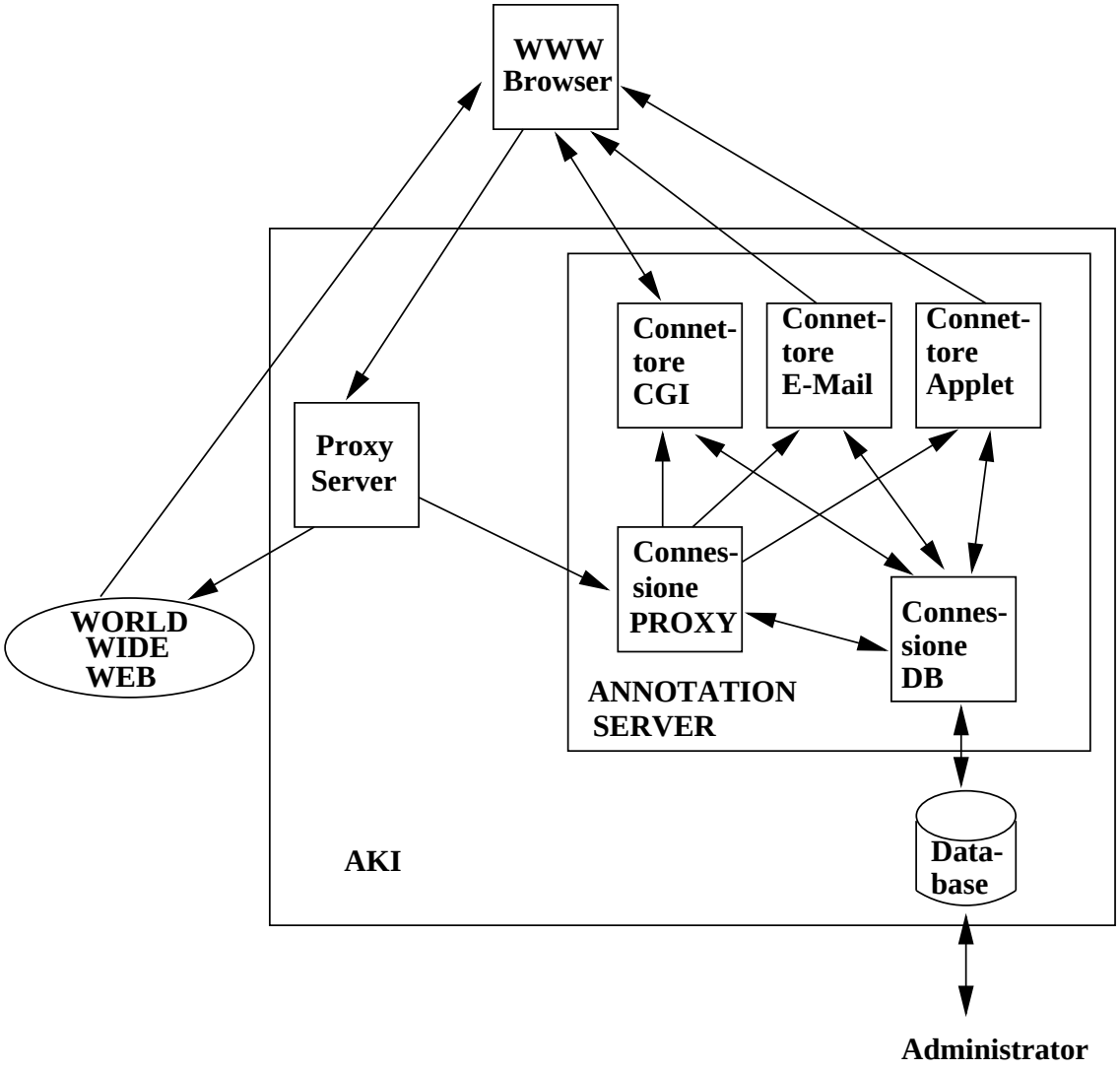

Figura 3.10: Architettura del sistema AKI.

3.6.3 Conclusioni

Il sistema AKI è stato progettato con l'aiuto del linguaggio *UML (Unified Modeling Language)* [Booch and Rumbaugh 1997], [Fowler 1997] ed implementato utilizzando il linguaggio Java.

L'utilizzo degli esistenti sistemi di annotazione, di cui nelle sezioni precedenti sono riportati alcuni esempi, richiede l'impiego di specifici *Browsers* o perlomeno di ampliare le funzionalità di quelli già esistenti. Tutto ciò inevitabilmente riduce l'accettazione di tali sistemi da parte degli utenti finali. Il sistema AKI, e la sua architettura riporatata in figura 3.10 lo dimostra, è stato realizzato per ridurre notevolmente tali problemi adottando soluzioni *standard* che favoriscono la portabilità dell'intero sistema.

Infine il sistema AKI prevede dei meccanismi per l'organizzazione degli utenti in gruppi,

favorendo l'istaurarsi di un lavoro cooperativo nell'utilizzo dei documenti distribuiti su tutto il *World Wide Web*.

Questo capitolo ha lo scopo di mostrare a grandi linee i principi generali che caratterizzano i sistemi, attualmente esistenti, a supporto dell'annotazione digitale. Le soluzioni ai vari problemi sono molto diverse e presentano vantaggi e svantaggi che devono essere accuratamente considerati in correlazione alle caratteristiche del particolare dominio applicativo.

Sulla base di tali concetti è effettuata la progettazione del sistema a supporto dell'annotazione digitale di documenti multimediali presentato nei capitoli successivi di tale testo.

Capitolo 4

Personalizzazione nelle Biblioteche Digitali

Lo scopo di una biblioteca digitale è sostanzialmente quello di migliorare lo sfruttamento dell'universo di informazioni che globalmente circolano sulla rete dando la possibilità all'utente di reperire informazioni specifiche necessarie allo svolgimento del suo compito.

In altre parole una biblioteca digitale è il software necessario per unire il mondo della rete, caratterizzato da un'immensa quantità di informazioni, ed il mondo dell'utente che invece necessita di particolari e ben precise informazioni per lo svolgimento del suo compito.

Uno dei requisiti principali dei documenti digitali consiste nel fatto che essi possono essere copiati con estrema rapidità e semplicità; in tal modo ogni utente ha la possibilità di avere copie personali dei documenti che desidera i quali possono essere manipolati e rielaborati in base alle necessità dello specifico utente. Questo è un aspetto importante del principio di personalizzazione che è alla base e che giustifica le biblioteche digitali.

Scopo di tale capitolo è quello di dare una definizione generale del settore delle biblioteche digitali, mettendo in evidenza gli aspetti di cooperazione e personalizzazione che hanno guidato la progettazione del sistema per l'annotazione di documenti multimediali descritto nei capitoli successivi di tale testo.

In una prima parte è descritto a grandi linee il settore delle biblioteche digitali cercando di metterne in evidenza le caratteristiche generali e le motivazioni che stanno alla base del suo sviluppo. Segue la presentazione dei concetti di personalizzazione e di cooperazione mediante la spiegazione dei criteri in base ai quali si può costruire una struttura che consenta il raggiungimento di tali obiettivi. In tale contesto viente presentato il progetto WEL (*Warburg Electronic Libraries*, sezione 4.2.2) che è la sede naturale in cui è stato sviluppato il sistema a supporto delle annotazioni presentato in tale testo. Lo spirito di tale capitolo è quindi descrittivo, con lo scopo, non secondario, di inquadrare in un contesto generale ma ben definito e ricco di contenuti un argomento specifico quale è il meccanismo di annotazione.

4.1 Evoluzioni nell'area delle Biblioteche Digitali

Le biblioteche digitali si sono sviluppate notevolmente quando la tecnologia dei *computers* e delle reti di calcolatori è migliorata rapidamente incrementando e favorendo una globale accet-

tazione delle informazioni digitali e dei servizi di comunicazione ad esse connessi.

La prima conseguenza di tale processo è rappresentata dal fatto che ora una grande quantità di informazioni è scambiata tra la comunità mondiale dei fornitori di informazioni ed i consumatori delle informazioni medesime. Tale potenzialità rivoluzionerà completamente e radicalmente tutte le attività di manipolazione e di gestione delle informazioni.

Le biblioteche digitali sono pensate come uno strumento attraverso il quale è possibile strutturare lo spazio globale delle informazioni [Li *et al.* 1996], migliorando così l'uso delle informazioni stesse.

In tale logica le biblioteche digitali devono rispettare la tradizione delle esistenti biblioteche fisiche e trascendere nello sviluppo di nuovi e più ampi ambiti di ricerca al fine di arricchire le funzionalità offerte dalle biblioteche tradizionali sfruttando i vantaggi offerti dal processo di digitalizzazione.

Infatti il campo emergente delle biblioteche digitali racchiude assieme argomenti e questioni dibattute in molti settori di ricerca moderni. Tra questi è possibile comprendere il settore di ricerca delle basi di dati, quello che si occupa dello studio delle modalità di trattamento delle informazioni multimediali ed infine anche il settore delle telecomunicazioni, nell'ambito del quale vengono elaborate tecniche sempre più innovative per il trasferimento delle informazioni digitali attraverso la rete.

Ci sono molte parole che indicano concetti correlati alle biblioteche digitali: *Multi-media Database* [Woelk and Kim 1987], *Information Mining, Information Warehouse, Information Retrieval, On-line Information Repositories, Imaging Database, World Wide Web* [Nickerson 1992], [Hahn and Stout 1994] e *WAIS* [Hahn and Stout 1994] (*Wide Area Information Service*). Questa lista di termini implica innumerevoli concetti che si sovrappongono a vicenda e che individuano campi di ricerca tra loro correlati.

Per questo motivo non è assolutamente semplice riuscire a definire in schemi rigidi e precisi l'emergente settore delle biblioteche digitali. Molti ricercatori sono portati a pensare che il settore delle biblioteche digitali sia una naturale crescita di altre discipline. Il campo delle biblioteche digitali però viene notevolmente limitato se considerato come un semplice sottoassieme di altri argomenti di ricerca.

Per comprendere e focalizzare appieno il problema, il settore delle biblioteche digitali deve essere analizzato da un punto di vista generale cercando di comprendere quali sviluppi ed a quali vantaggi potrebbe portare il processo di digitalizzazione delle biblioteche tradizionali.

Da quanto è stato detto si può capire che le biblioteche digitali si differenziano da quelli che sono i tradizionali *Database*.

Le biblioteche digitali differiscono dai *Database* in tre aspetti fondamentali:

- le biblioteche digitali accedono a informazioni eterogenee rappresentate da vari modelli relativi a diversi mezzi di comunicazione;
- l'architettura delle biblioteche digitali è dinamica e consente una costruzione fortemente personalizzata di informazioni e di servizi;
- gli utenti delle biblioteche digitali lavorano con processi di contesto i quali sono tracciati ed elaborati dall'ambiente di biblioteca.

Le biblioteche digitali sono necessarie anzitutto per fornire alle biblioteche tradizionali nuovi e più perfezionati servizi ma anche per ridurre i costi di fornitura di tali servizi in un settore che si trova in piena crisi.

I sogni, di oltre un decennio, di biblioteche basate sui *computers* [Bush 1945] sono ora realizzabili concretamente.

4.1.1 Perché le Biblioteche Digitali

Attualmente le biblioteche tradizionali sono di fronte ad un doppio e simultaneo problema: la loro impossibilità ad acquistare nuovo materiale[1] e la loro incapacità a mantenere materiale vecchio.

L'approccio elettronico può, in generale, essere un'adeguata risposta ad entrambi tali problemi; l'elettronica può essere una buona soluzione al problema del deterioramento in quanto introduce più elevati gradi di sicurezza ma soprattutto, attraverso il trattamento elettronico delle informazioni, si riduce notevolmente il costo relativo all'immagazzinamento ed alla gestione delle risorse informative.

A ciò si aggiunge il fatto che attualmente viene realizzato nuovo materiale bibliografico, sempre e comunque, in forma digitale e quindi subito disponibile per essere gestito per mezzo di una biblioteca digitale.

Per tali ragioni la digitalizzazione delle informazioni può essere sicuramente vista come la soluzione a numerosi problemi; inoltre l'elettronica può offrire enormi vantaggi alle biblioteche. Tra questi si può considerare il miglior accesso al materiale digitale, la più facile memorizzazione delle informazioni e la possibilità di estendere notevolmente le attuali collezioni delle biblioteche tradizionali.

Non è difficile immaginare come sia notevolmente semplice e veloce effettuare delle ricerche su materiale elettronico. Il trattamento elettronico delle informazioni fornisce inoltre notevoli altri vantaggi; ad esempio l'obiquità è un chiaro limite degli oggetti fisici. Una singola copia elettronica può essere condivisa da un elevato numero di postazioni e quindi da un elevato numero di utenti.

Un altro importante vantaggio è la conservazione. Le informazioni digitali possono essere copiate facilmente e senza errori. In questa logica la preservazione delle informazioni digitali non consiste nell'avere un singolo oggetto mantenuto permanentemente sotto controllo bensì nell'abilità di fare copie multiple assumendo che, da un punto di vista statistico, almeno una sopravviverà.

L'immagazzinamento digitale permette anche alle biblioteche di espandere notevolmente la quantità di materiale che esse possono offrire ai loro utenti.

Uno degli aspetti di fondamentale importanza delle biblioteche digitali è però costituito dalla possibilità di un lavoro cooperativo tra diverse librerie distribuite in luoghi geografici diversi. In tal modo si può avere l'impressione di utilizzare un'unica grande biblioteca costituita in realtà da una serie di biblioteche distribuite sul territorio.

[1] Ciò è dovuto al fatto che trovare finanziamenti per la gestione delle biblioteche è un compito sempre più difficile nella società moderna, caratterizzata da altri e più gravi problemi.

L'informazione del futuro può essere pensata come un oceano; il compito delle biblioteche non sarà più quello di doversi preoccupare di alimentare questo oceano con nuova acqua bensì quello di fornire una barca con cui poter navigare.

4.1.2 Perché le Biblioteche Digitali sono delle Biblioteche

Una delle domande che è lecito porsi è perché una biblioteca digitale, od una biblioteca elettronica, od una biblioteca virtuale può essere chiamata o deve essere chiamata "Biblioteca"?

Una biblioteca digitale potrebbe essere indicata semplicemente come un sistema di informazioni digitali, o come un sistema di pubblicazioni digitali; entrambe le alternative sono possibili.

Ma tali alternative non sono state scelte; invece biblioteca è stato il termine scelto. E questa scelta non è stata fatta dai bibliotecari bensì dagli studiosi di *computers* ed informazione che si sono occupati principalmente dello sviluppo di sistemi per la comunicazione elettronica nei tre decenni precedenti.

Per rispondere alla domanda sopra posta occorre analizzare i tre aspetti che stanno alla base della definizione delle tradizionali biblioteche come una *collezione di risorse informative posizionate in un determinato luogo* [Miksa and Doty 1994].

L'idea di **collezione** deve essere esaminata come punto di partenza in quanto implica limiti pragmatici e necessari; a ciò segue il concetto di risorse **informative** ed infine la nozione di **luogo** inteso per lo più secondo un'accezione logica che non fisica.

Sulla base di questi concetti devono essere costruite le biblioteche digitali le quali devono fornire una serie di metodi flessibili per la gestione di risorse informative digitali in spazi condivisi e distribuiti.

Tutto ciò ovviamente richiede l'utilizzo della rete Internet quale infrastruttura a supporto dello scambio di informazioni digitali.

4.1.3 Requisiti generali ed architetture alla base delle biblioteche digitali

Nello sviluppo di sistemi a supporto delle biblioteche digitali il primo passo consiste nell'individuazione dei requisiti richiesti e delle architetture da utilizzare.

Scopo di tale sezione non è certo quello di trattare in modo esaustivo l'argomento bensì quello di individuare le questioni di maggior importanza per quanto riguarda i concetti di cooperazione e personalizzazione , i quali costituiscono uno dei motivi che hanno portato allo sviluppo del sistema per l'annotazione presentato in tale testo.

Un servizio completo di biblioteca deve contenere molti componenti, l'istallazione dei quali seleziona un sottoinsieme di elementi da cui ogni utente può attingere l'informazione voluta.

Un primo passo da fare consiste nella classificazione degli elementi delle biblioteche lungo due assi; in primo luogo gli elementi possono essere classificati in dati, metadati e servizi mentre sul secondo asse possono essere classificati in elementi delle biblioteche fisiche e nuovi elementi delle librerie digitali come mostra la figura 4.1.

	DATI	METADATI	SERVIZI
Tabulazione delle entita' delle librerie fisiche	LIBRI GIORNALI RIVISTE ILLUSTRATE	INDICI STATICI CLASSIFICAZIONI ORGANIZZAZIONE SPAZIALE	ACQUISIZIONE DATI PROPOSTA RISORSE AIUTO LOCAZIONE RISORSE
Nuove entita' delle librerie digitali	IPERNOVELLE VISUALIZZAZ. SCIENTIFICHE PROGRAMMI DI CALCOLO	INDICI DINAMICI STRUTTURE PERSONALIZ. ANNOTATIONS	RICERCA NEL TESTO PRESENT. PERSONALIZ. RECUPERO TRAMITE AGENTI

Tabella 4.1: Classificazione degli elementi di una biblioteca digitale.

I requisiti riportati nelle celle della tabella sopra visualizzata pongono dei problemi che sono riassunti nella figura 4.2. Le due tabelle (4.1 e 4.2) sono tratte da [Schnase *et al.* 1994].

	DATI	METADATI	SERVIZI
Tabulazione delle entita' delle librerie fisiche	COSA VIENE TRADOTTO?	COME TRADURRE METADATI CHE SONO INDIPENDENTI DAI DATI FISICI?	COME FORNIRE STRUMENTI PER IL COINVOLGIMENTO UMANO ?
Nuove entita' delle librerie digitali	COME CONSIDERARE LA CONTINUA E RAPIDA EVOLUZIONE DI NUOVI TIPI DI DATI?	COME ASSICURARE CONSISTENZA NELLA SEPARATA GESTIONE DEI METADATI?	COME DISTRIBUIRE L'ELABORAZIONE ?

Tabella 4.2: Problemi correlati agli elementi di una biblioteca digitale.

Un sistema a supporto delle biblioteche digitali potrebbe essere pensato come un meccanismo di mediazione che consente l'interazione tra persone e sistemi computerizzati. La figura 4.1 mostra chiaramente le relazioni e le interazioni tra le parti di una biblioteca digitale ed alcune persone e sistemi esterni alla biblioteca[2]. Le risorse di calcolo in figura 4.1 sono state partizionate in risorse *Server* ed in risorse *Client*.

Il cliente accede alle risorse gestite dalla biblioteca digitale attraverso un *Client*; il *Client* consente fondamentalmente il recupero delle informazioni desiderate. La gestione della biblioteca digitale avviene invece sul *Server* a cui hanno accesso il bibliotecario, ovvero colui che è responsabile direttamente della gestione del sistema, e l'editore che fornisce nuovo materiale bibliografico in formato digitale.

Nella realtà le relazioni esistenti sono molto più complicate rispetto a quanto mostrato in figura 4.1. Inoltre i sistemi *Client* e *Server* di calcolo possono essere ulteriormente suddivisi.

Ognuno può essere pensato come consistente in tre parti: il *back-end*, il *middle-end* ed il *front-end*; sia il *back-end* che il *front-end* sono meccanismi che definiscono l'interfaccia tra il

[2] Anche la figura 4.1 è tratta da [Schnase *et al.* 1994].

sistema stesso e le entità esterne. Il sistema *front-end* fornisce servizi a *Clients* esterni mentre il *back-end* è realizzato con servizi offerti da *Servers* esterni. Il *middle-end* invece è un mezzo di comunicazione intermedia tra *back-end* e *front-end*.

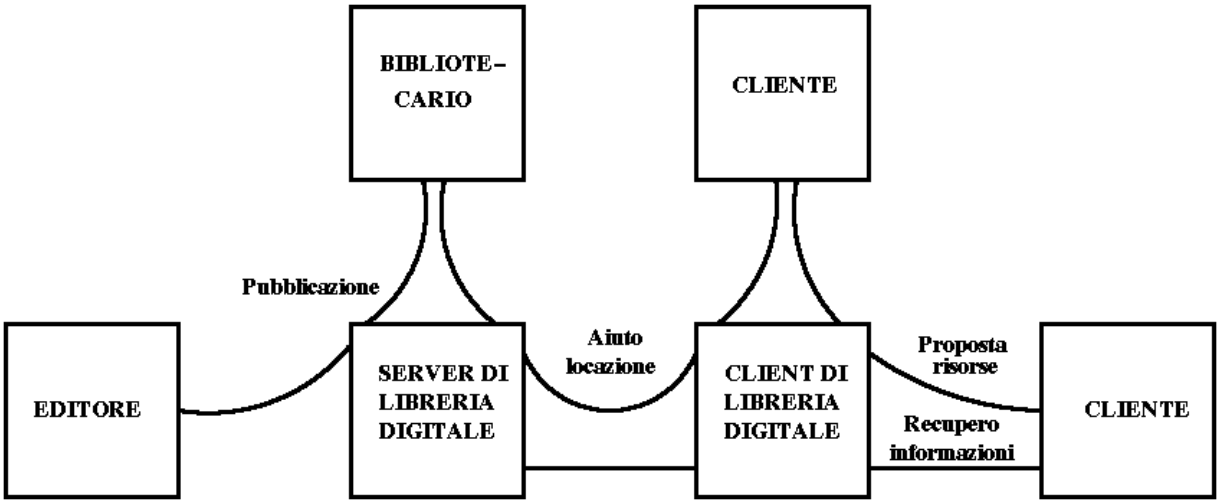

Figura 4.1: Ruoli e relazioni in un sistema a supporto di biblioteche digitali.

Infine occorre effettuare una considerazione che sta alla base dei problemi trattati nel testo ovvero che gli attuali *Web Clients* disponibili commercialmente consentono di ottenere solamente un piccolo grado di personalizzazione.

I *Web Clients* devono essere arricchiti con un maggior numero di strumenti che consentano una facile personalizzazione dei dati con il rispetto sia della loro presentazione che del loro accesso.

In tale ottica si inseriscono i sistemi a supporto delle annotazioni.

4.1.4 Il futuro delle Informazioni e delle Biblioteche Digitali

Uno dei più importanti vantaggi offerti da un sistema che supporta le biblioteche digitali è la possibilità di effettuare rapide ed efficaci operazioni di ricerca. Quando tutti i testi completi sono disponibili in una base di dati ogni parola o frase può essere trovata rapidamente e con estrema facilità.

Per tale ragione gli utenti delle biblioteche digitali possono usufruire di un set di servizi molto più evuluti ed efficienti rispetto a quelli forniti dalle biblioteche tradizionali. Inoltre gli oggetti digitali hanno una grandissima proprietà ovvero quella di poter essere copiati senza che nessun errore venga commesso; quindi non vi è più alcuna preoccupazione circa il deterioramento dei mezzi fisici, tanto che le copie dei documenti possono essere distribuite con facilità all'interno della biblioteca.

La cosa certa è che in futuro le persone spenderanno sempre più in informazione rispetto al

presente; la distribuzione di informazione diventerà un *business* con grandi economie di scala e con gravi conseguenze per la stabilità del mondo economico.

In un tale scenario le biblioteche dovranno collaborare per poter sopravvivere; esse dovranno fornire agli utenti uno strumento per la navigazione attraverso il mare dell'informazione.

4.2 Aspetti di Personalizzazione e Cooperazione

Sin dalla sua nascita il World Wide Web è stato pensato come un sistema per dare alle persone la possibilità di cooperare e di scambiare informazioni attraverso una rete estesa su una grande e distribuita area geografica. Questo risultato è stato raggiunto attraverso la scrittura di documenti multimediali, contenenti immagini, testi e suoni, da parte di utenti che rendono disponibili questi documenti al pubblico, memorizzandoli su *Servers* che sono distribuiti geograficamente in tutto il mondo.

In un tale meccanismo ogni documento può contenere *links* a documenti che si trovano nello stesso sito ma può anche aver riferimenti a documenti che si trovano su di un qualsiasi *Server* che fa parte della grossa infrastruttura a supporto dello scambio di informazioni che è la rete Internet.

Di seguito verranno presentati i principi generali che stanno alla base di una struttura che consenta cooperazione e personalizzazione; in tale contesto verrà presentato l'approccio adottato presso la Techmische Universität Hamburg-Harburg al fine della progettazione delle biblioteche digitali e verrà descritto, a grandi linee, il progetto WEL-*Warburg Electronic Libraries* (sezione 4.2.2) nel cui contesto si inserisce il lavoro contenuto in tale testo.

Si ha cooperazione nel momento in cui più persone sono coinvolte in un compito che comporta le scrittura di uno stesso documento in una via cooperativa e di collaborazione. L'obiettivo finale è quello di scrivere e produrre un documento complesso e ben strutturato e non semplicemente scrivere vari pezzi di contenuti informativi interconnessi tra di loro da dei *links*, come in un mosaico.

Tale aspetto di cooperazione assume rilevante importanza nell'ambito del *World Wide Web* in cui sono state sviluppate innumerevoli soluzioni, relative ad un approccio cooperativo, che possono essere applicate ad un generale concetto di *editing* cooperativo.

4.2.1 Principi generali

L'essenza delle biblioteche digitali può essere riassunta in tre punti fondamentali [Schmidt *et al.* 1997]:

- Il contenuto di una biblioteca digitale è rappresentato da due unità informative:

  a) a livello di base ci sono i cosiddetti *Information Tokens* che rappresentano l'informazione fornita dalla rete;

  b) il contenuto informativo è aggiunto a questo *Tokens* da un *Information Artifacts* che si trova sopra di esso e che ha lo scopo principale di soddisfare le particolari richieste del consumatore delle informazioni realizzando in tal modo una personalizzazione delle informazioni [Röscheisen *et al.* 1994], [Röscheisen *et al.* 1995b];

- I servizi richiesti per la costruzione dell'*Artifacts* e per il suo utilizzo;

- I processi per la costruzione e l'uso dell'*Artifacts* sono essi stessi risorse informative utilizzabili dall'*Information Artifacts*; per lo sfruttamento di tali processi informativi le biblioteche digitali impiegano un avanzato *Tracing Enviroments*.

4.2.2 WEL-*Warburg Electronic Library*

In tale ampio contesto delle biblioteche digitali trova la sua sede naturale il progetto WEL (*Warburg Eletronic Library*) con lo scopo di costruire una biblioteca digitale basata su icone, testi e dati.

Il *Warburg Electronic Library Project* (WEL) è iniziato nell'ambito della cooperazione interdisciplinare [Niederée *et al.* 1996] tra il gruppo di ricerca della *TU Hamburg-Harburg* e l'*Art History department* dell'università di Hamburg.

L'obiettivo principale di tale progetto consiste nell'esame, sviluppo ed applicazione delle biblioteche digitali per scopi di ricerca storica.

Il dominio applicativo della storia dell'arte è dominato da immagini arricchite con del testo ed artefatti multimediali come filmati o documenti audio.

La ricerca nel campo della storia dell'arte si focalizza sull'esame di tali immagini, sull'identificazione di icone rappresentative eventi e/o persone e sulla loro classificazione in base a determinate logiche.

Nella gestione e manipolazione delle immagini il processo automatico di trattamento delle stesse gioca un ruolo secondario e poco significativo. Molte informazioni devono essere aggiunte dalle persone in un processo di arricchimento dei contenuti che è di fondamentale importanza.

In una tale biblioteca digitale gli *Information Tokens* sono costituiti da fotografie o dipinti raffiguranti regnanti a cui è associato del testo. Tali *Information Tokens* vengono creati ed inseriti nella grande quantità di informazioni attraverso un processo di *scanning* od equivalenti procedure di digitalizzazione; altra risorsa di *Tokens* è rappresentata dall'inserimento di testo. Le immagini vengono salvate come vettori di bit in formati come JPEG o GIF.

Gli *Information Tokens* sono presi come punto di partenza per il processo di costruzione di artefatti descritto in precedenza. La *Warburg Electronic Library* mantiene una larga collezione di tali artefatti assieme ai servizi per la loro memorizzazione persistente, la trasmissione, il recupero e la presentazione.

Cooperazione e Personalizzazione

Le biblioteche digitali quindi forniscono materiale multimediale per gruppi di consumatori di informazione i quali possono personalizzare il loro contenuto tramite riferimenti personali. Nell'ambito del progetto WEL è ancora in atto lo sviluppo di un ambiente che consenta la personalizzazione e la cooperazione; in tale contesto si inserisce il lavoro descritto in tale testo in quanto le annotazioni costituiscono un aspetto fondamentale dei concetti di personalizzazione e di lavoro cooperativo [Davis and Huttenlocher 1995].

Attraverso l'annotazione di documenti multimediali è possibile aggiungere notizie e commenti personali che possono essere resi disponibili a tutti gli utenti del sistema oppure consultabili solo in forma privata. Inoltre un sistema per l'annotazione deve prevedere meccanismi di organizzazione logica degli utenti in gruppi all'interno dei quali è possibile realizzare un lavoro cooperativo.

Nel contesto del progetto WEL appare chiara la necessità di un sistema a supporto delle annotazioni a documenti multimediali, costituiti sia da testo che da immagini.

In conclusione si può affermare che l'aspetto centrale delle biblioteche digitali è senza dubbio costituito dal processo di costruzione di artefatti che portano ad un arricchimento del contenuto informativo degli oggetti presenti sulla rete (*Information Tokens*) consentendo quindi la personalizzazione delle informazioni e lo svolgimento di un lavoro cooperativo.

4.3 Le Annotazioni e le Biblioteche Digitali

Scopo di tale sezione è quello di identificare le ragioni in base alle quali la procedura di annotazione di documenti multimediali risulta essere di fondamentale importanza al fine della realizzazione dei meccanismi di cooperazione e personalizzazione nell'ambito delle biblioteche digitali.

Generalmente le persone annotano i libri nel momento in cui, nel corso della lettura, assumono un ruolo attivo arricchendo il contenuto del documento con informazioni aggiuntive e considerazioni personali.

Annotare un libro è certamente un'operazione di estrema utilità anche se tale procedura non può essere applicata concretamente ai libri che si vanno a prendere nelle biblioteche; collezioni di materiale digitale possono liberare i lettori da questi vincoli. A dispetto di ciò le biblioteche digitali sono strumenti che non consentono elevata flessibilità per quanto riguarda il meccanismo di annotazione offerto ([Duffy 1993], [Moulthrop 1992], [Marshall 1997]).

Queste sono le motivazioni alla base dello sviluppo del sistema per l'annotazione di documenti multimediali proposto in tale testo. In tale direzione occorre definire in modo formale quali sono i requisiti che deve possedere un sistema per l'annotazione digitale, affinché possano essere realizzati gli aspetti di personalizzazione e cooperazione che costituiscono un arricchimento del concetto di biblioteca digitale.

Annotazioni e Personalizzazione

La procedura di annotazione di un documento costituisce sicuramente un'operazione che porta il lettore a personalizzare il contenuto informativo del documento medesimo. L'annotazione infatti potrebbe consistere in un commento attraverso cui il lettore esprime considerazioni personali in relazione ai concetti espressi nel testo o nelle immagini riportate nel documento oggetto di lettura.

Altra possibilità potrebbe essere quella che il lettore selezioni alcune idee riportate nel documento, eventualmente evidenziandole con sottolineature o con simboli grafici di vario genere.

Anche questo è un aspetto di personalizzazione in quanto riflette la modalità attraverso cui il lettore interpreta il testo ritenendo alcuni concetti più importanti di altri.

Ed ancora la possibilità che il lettore evidenzi e sintetizzi solamente alcuni concetti riportati nel documento in funzione dell'ambito specifico di lavoro in cui si trova ad operare; ciò significa che utenti diversi, operanti in campi diversi ed aventi una formazione culturale e professionale diversa, possono leggere un documento da svariati punti di vista mettendone in risalto alcune idee piuttosto che altre.

Appare chiaro quindi che avere la possibilità di annotare un documento significa avere l'opportunità di aggiungere qualcosa di proprio e di personale al documento oggetto di lettura che, in tal modo, diviene unico ed irripetibile.

Annotazioni e Cooperazione

La cooperazione è un concetto di estrema importanza nel momento in cui più persone sono impiegate nello svolgimento di un determinato lavoro per il conseguimento di obiettivi comuni. In tal caso si ha il concetto di gruppo di utenti che lavorano per il conseguimento di uno scopo ben definito.

In tale contesto avere la possibilità di definire annotazioni e di renderle disponibili a tutti i membri del gruppo rappresenta senza dubbio un aspetto che favorisce la cooperazione nello svolgimento del lavoro comune.

Sulla base di questa logica risultano essere estremamente utili i meccanismi di notificazione attraverso cui i membri del gruppo vengono portati a conoscenza, ad esempio, di nuove annotazioni associate ad un determinato documeno e riguardanti una particolare tematica. Ciò significa che i meccanismi di notificazione sono un efficace strumento per la comunicazione interna, tra membri di un determinato gruppo, e costituiscono il presupposto per la realizzazione di un lavoro cooperativo.

Infine la possibilità di effettuare annotazioni riferite ad altre annotazioni favorisce lo sviluppo di comunicazioni multiutente che costituiscono sicuramente un aspetto rilevante del principio di cooperazione. A tal riguardo un utente potrebbe fare un'annotazione, riguardante il contenuto di un documento, e metterla a disposizione dei componenti del proprio gruppo di lavoro. Questi ultimi a loro volta, disponendo del documento originale e dell'annotazione, potrebbero fare altre considerazioni sotto forma di annotazioni all'annotazione già esistente, nell'istaurarsi quindi di una comunicazione cooperativa che porta ad un lavoro sicuramente più efficace ed efficiente.

Tale capitolo rappresenta la contestualizzazione del meccanismo di annotazione nell'ambito delle biblioteche digitali, con particolar riguardo ai principi di personalizzazione e di cooperazione che tale meccanismo consente di realizzare.

Sulla base di tali concetti generali è realizzato il sistema per l'annotazione di documenti multimediali in spazi condivisi descritto nei capitoli successivi di tale testo.

Parte II

Sistema di annotazione a supporto della cooperazione e personalizzazione

Capitolo 5

Un modello per il sistema di annotazione

Una rigorosa ed attenta fase di modellizzazione è sempre il presupposto basilare alla buona riuscita ed al successo di un progetto. La costruzione del modello aiuta a comprendere a fondo il mondo che si deve rappresentare fornendo i requisiti indispensabili alla successiva fase di implementazione del prototipo.

In questo capitolo è riportata una dettagliata descrizione del modello elaborato per lo sviluppo di un sistema a supporto del meccanismo di annotazione. In fase di modellizzazione è stato utilizzato il linguaggio UML (*Unified Modeling Language*) di cui è riportata una breve descrizione nel capitolo 6. Al fine di non appesantire l'esposizione con una lunga serie di diagrammi si è deciso di fare un'appendice (appendice B) in cui sono raccolti tutti gli schemi principali realizzati in fase di progettazione. Si consiglia quindi anche una lettura di tale appendice per meglio comprendere i dettagli del modello proposto.

I contenuti di tale capitolo sono strettamente correlati ai concetti esposti nel capitolo 7 relativo alla descrizione dell'architettura del sistema e delle funzionalità del prototipo.

5.1 Dominio applicativo

La corretta definizione del dominio applicativo costituisce il primo passo che deve essere compiuto in fase di modellizzazione. Ciò che viene proposto in tale testo è un sistema che offre la possibilità agli utenti di effettuare delle annotazioni a documenti presenti sul WWW.

A tal proposito lo scenario è molto vario; ciò che deve essere chiaro, e considerato con particolare attenzione, è che le annotazioni devono rappresentare uno strumento che consenta all'utente di realizzare i principi di **cooperazione** e di **personalizzazione** a lungo discussi nel capitolo 4.

A tal proposito il sistema deve offrire la possibilità di creare delle gerarchie di annotazioni in cui cioè annotazioni annotano altre annotazioni; questo permette lo svilupparsi di discussioni tra gli utenti che possono quindi cooperare nello svolgimento del loro lavoro. Inoltre deve essere possibile riferire le annotazioni all'intero documento oppure a sue parti interne, quali porzioni di testo oppure immagini in esso contenute. Quest'ultimo aspetto aumenta la flessibilità del sistema fornendo un maggior numero di gradi di libertà agli utenti. Fatte tali considerazioni appare

evidente quindi che un'annotazione dovrà poter essere costruita come un normale documento HTML contenente testo, immagini e riferimenti ad altri documenti.

Altro aspetto principale del modello di annotazione è l'organizzazione degli **utenti**; gli utenti sono organizzati in gruppi che si possono nominare con la dizione di **progetti**. Ogni progetto è costituito da un gruppo di persone legate da interessi prevalentemente comuni; il progetto è caratterizzato da obiettivi da raggiungere e da un certo tempo entro cui deve essere portato a termine. L'aspetto interessante è che i progetti possono essere organizzati in una gerarchia di sottoprogetti. Una persona che è membro di un determinato progetto indirettamente appartiene anche a tutti i progetti che si trovano al di sopra di esso nella struttura gerarchica. Ogni progetto possiede un **Project Administrator** che è colui che ha creato il progetto e che dovrà quindi occuparsi di gestirlo.

Il sistema deve inoltre essere dotato di un componente attivo (*Annotation Server*) in grado di realizzare un meccanismo di **notificazione** [Andreessen 1993]. La notificazione consiste essenzialmente nella spedizione di *e-mails*[1] a determinati utenti quando nel sistema si verificano particolari eventi nel rispetto di precise condizioni. La logica è la stessa di quella che sta alla base delle cosiddette *ECA-Rules* [Ceri *et al.* 1997], [S.Ceri *et al.* 1996], [Ceri and Ramakrishnan 1996] in base alle quali si possono identificare i seguenti componenti:

- gli **eventi** che scatenano il meccanismo di notificazione sono sostanzialmente la creazione di una nuova annotazione, la sua cancellazione, l'inserimento e la cancellazione di un utente da un gruppo di progetto;

- le **condizioni** che devono essere soddisfatte riguardano le modalità in base alle quali le annotazioni sono sate realizzate e la locazione degli utenti nell'albero gerarchico dei progetti;

- l' **azione** che viene eseguita è la spedizione di *e-mails* agli utenti che rispettano le condizioni stabilite.

Per realizzare tale meccanismo è quindi necessario definire degli **Access Modifiers** per le annotazioni i quali devono essere specificati dall'autore al momento della creazione.

Deve infine esistere una persona particolare, il **System Administrator**, il quale ha il compito di gestire il sistema; egli è l'unico utente che ha la possibilità di cancellare le annotazioni fatte da un qualsiasi utente in base a determinati criteri.

Una classificazione delle componenti attive che fanno parte dell'*Annotation Server* è riportata nella sezione 5.9; nella sezione 8.2 sono inoltre riportate, a titolo esemplificativo, le implementazioni di alcune regole attive identificabili nel modello astratto.

Tutto ciò è pensato alla luce dei concetti di personalizzazione e di cooperazione; il meccanismo di annotazione è infatti un presupposto basilare al fine della personalizzazione dei documenti multimediali. I meccanismi di ricerca delle annotazioni, la possibilità di effettuare annotazioni ad annotazioni, l'organizzazione gerarchica dei progetti ed i meccanismi di notificazione rappresentano invece aspetti importanti del principio di cooperazione.

[1] Tale meccanismo di notificazione viene detto *off-line* ed è caratterizzato dal fatto di essere asincrono.

Questi sono, sinteticamente, i requisiti richiesti al sistema sviluppato; le sezioni successive di tale capitolo descrivono in maniera dettagliata ogni aspetto del modello proposto mettendo in evidenza le soluzioni adottate per risolvere i vari problemi che si sono presentati nel corso dello sviluppo. Si consiglia di fare riferimento all'appendice A in cui è riportato il Glossario, al fine della comprensione di particolari termini tecnici utilizzati nel corso dell'esposizione.

5.2 *Use Cases*

Gli *use cases* rappresentano sostanzialmente le modalità attraverso le quali l'utente interagisce con il sistema ottenendo i risultati desiderati. La corretta identificazione degli *use cases* è un passo preliminare allo sviluppo del modello, ma condiziona enormemente il cammino che si intraprenderà in fase di progettazione.

Ancora prima della definizione degli *use cases* occorre individuare gli ***user goals*** (obiettivi dell'utente del sistema) ovvero ciò che l'utente desidera realmente dal sistema di annotazione. Di seguito è riportato un elenco degli *user goals* individuati ed una breve spiegazione del significato di ognuno di essi:

- Creazione di una nuova annotazione
- Lettura di un'annotazione
- Cancellazione di un'annotazione
- Ricerca di un'annotazione
- Notificazione
- Definizione di un nuovo progetto
- Chiusura di un progetto esistente
- Ingresso in un gruppo di progetto
- Uscita da un determinato progetto
- Gestione dei temi di interesse

Questi sono in generale gli obiettivi che un utente del sistema di annotazione desidera raggiungere. Prima di proseguire con una descrizione più dettagliata di ogni *user goal* è necessaria una considerazione di estrema importanza e che può essere facilmente individuata da un'attenta analisi della lista sopra riportata: tra gli *user goals* non è prevista la possibilità che l'utente modifichi un'annotazione già esistente. Questo è un presupposto allo sviluppo del sistema in quanto si è deciso di non dare la possibilità nemmeno all'autore dell'annotazione di modificare l'annotazione medesima. Questa scelta è stata dettata dal fatto che il sistema rende possibile la creazione di una gerarchia di annotazioni tramite annotazioni fatte ad altre annotazioni e non solo a documenti. Se un utente modificasse quindi un'annotazione già effettuata potrebbe darsi

che la gerarchia al di sotto di essa non avrebbe più significato di esistere e questo porterebbe ad un'inconsistenza delle informazioni presenti nel sistema. Ad esempio se un utente effettua un'annotazione riferita ad un documento scientifico riportando una propria idea è possibile che altri utenti del sistema esprimano proprie opinioni sotto forma di annotazioni all'annotazione originale; se a questo punto l'utente iniziale decidesse di portare delle modifiche alla sua idea potrebbe darsi che alcune considerazioni presenti nella gerarchia sottostante non abbiano più alcun significato perché riferite a concetti passati ed ora modificati.

Nel momento in cui un utente volesse modificare il pensiero che aveva precedentemente espresso con una certa annotazione potrebbe quindi fare una nuova annotazione riferita a quest'ultima retificandone il contenuto. Questa soluzione preserva la storia passata e crea il presupposto per l'evolversi di vere e proprie comunicazioni multiutente fornendo le basi per lo sviluppo di una struttura che consenta la cooperazione.

Fatta questa premessa doverosa e di estrema importanza è ora possibile procedere con una descrizione più dettagliata di ognuno degli *user goals*.

Creazione di una nuova annotazione: la creazione di una nuova annotazione è la funzionalità principale che il sistema deve offrire. L'utente deve essere in grado di creare annotazioni consistenti in testo e/o immagini le quali si riferiscono a documenti presenti sul *World Wide Web*, ad altre annotazioni oppure ad elementi interni a documenti ed annotazioni quali immagini e/o porzioni di testo. È possiblile quindi identificare due tipologie di annotazioni che l'utente può effettuare:

- Annotazioni posizionate in un punto preciso del documento
- Annotazioni riferite all'intero documento annotabile (documento od annotazione).

Lettura di un'annotazione: la lettura di un'annotazione è una funzionalità che è estremamente chiara e quindi non necessita ulteriori spiegazioni. L'utente deve poter leggere le annotazioni fatte a determinati documenti[2] nel rispetto delle logiche dei diritti d'accesso spiegate di seguito in tale capitolo.

Cancellazione di un'annotazione: il sistema deve offrire la possibilità di cancellare annotazioni già esistenti; tale operazione però non deve poter essere effettuata da ogni utente bensì solamente dal *System Administrator* che dovrà quindi decidere, in base a determinate logiche, quali annotazioni eliminare dalla base di dati.

Ricerca di un'annotazione: l'utente deve avere la possibilità di effettuare una ricerca delle annotazioni di suo interesse in base a criteri flessibili. Gli *use cases* riportano una definizione dettagliata dei criteri di ricerca offerti dal sistema.

Notificazione: questa è una funzionalità molto importante supportata dal sistema sviluppato. Ogni utente deve essere notificato, ovvero deve essere messo a conoscenza di "accadimenti" che si verificano nel sistema ed a cui è interessato. In genere gli eventi che possono

[2]Nel seguito del testo quando si utilizzerà il termine documento ci si riferirà sia a documenti veri e propri messi a disposizione dai *Servers Web* distribuiti in tutto il mondo che alle annotazioni, le quali sono esse stesse dei documenti.

scatenare il meccanismo di notificazione sono la creazione di una nuova annotazione, la cancellazione di un'annotazione, l'inserimento e la cancellazione di un utente dal gruppo di un determinato progetto.

Definizione di un nuovo progetto: ogni utente può creare uno o più progetti di cui diviene automaticamente l'amministratore. Il concetto di progetto verrà trattato di seguito in modo più esauriente; per ora si può pensare il singolo progetto come un gruppo di persone legate da interessi prevalentemente comuni e che lavorano per il conseguimento di determinati obiettivi.

Chiusura di un progetto esistente: il *Project Administrator* ha la possibilità di cancellare il proprio progetto dall'albero gerarchico dei progetti registrati nel sistema. Ciò è possibile solo se il progetto non possiede sottoprogetti; in caso contrario è necessario anzitutto cancellare i sottoprogetti e quindi chiudere il progetto desiderato.

Ingresso in un gruppo di progetto: un utente deve avere la possibilità di poter entrare a far parte di un determinato progetto. Solo l'amministratore del progetto può aggiungere altri componenti al suo gruppo.

Uscita da un determinato progetto: un determinato individuo può anche decidere di non far più parte di un determinato progetto; in questo caso è sempre l'amministratore del progetto ad avere i diritti di cancellare membri del proprio team.

Gestione dei temi di interesse: il concetto di tema o meglio di *Subject* verrà trattato specificatamente nella sezione 5.5.1. Ogni persona può avere dei temi a cui è interessata; ma anche i progetti e le annotazioni hanno dei soggetti. Il sistema deve offrire la possibilità di poter gestire tali *Subjects* ovvero la possibilità di modificarli, di aggiungerne dei nuovi o di cancellare quelli già esistenti .

Fatta la classificazione degli *user goals* occorre, come passo successivo, definire quali sono gli **Attori**[3] del sistema a supporto del meccanismo di annotazione. A tal proposito si possono individuare i seguenti ruoli:

Annotation Server: l'*Annotation Server* è un componente dell'intero sistema (per avere maggiori informazioni riguardanti l'architettura del sistema fare riferimento alla sezione 7.1). L'*Annotation Server* è un componente "reattivo"[4] in quanto realizza il meccanismo di notificazione oltre a tutte le attività di gestione della base di dati.

User: l'utente è una qualsiasi persona registrata nel sistema; più in dettaglio si possono individuare diversi ruoli che, all'interno del sistema, hanno dei diritti e quindi sono abilitati a svolgere compiti diversi.

[3]Per attore si intende chiunque richiede lo svolgimento di una certa funzionalità da parte del sistema di annotazione.

[4]Ciò significa che l'*Annotation Server* è in grado di eseguire determinate operazioni a seguito del verificarsi di particolari eventi che rispettano specifiche condizioni.

Annotation Reader: colui che legge un determinato insieme di annotazioni.

Annotation Author: colui che crea una determinata annotazione riferita ad un documento od ad un'altra annotazione.

Project Administrator: [5] tale attore rappresenta l'amministratore del progetto; il *Project Administrator* non coincide necessariamente con una figura fisica ma rappresenta sostanzialmente un ruolo attraverso cui vengono gestiti tutti gli aspetti relativi ad un determinato progetto.

System Administrator: [6] anche questo è un ruolo sostanzialmente amministrativo; il *System Administrator* è l'amministratore del *Workspace Public* nonché amministratore dell'intero sistema.

Definiti tali aspetti preliminari sono di seguito riportati alcuni degli *use cases* ritenuti più significativi rispetto ad altri, integrati con una breve spiegazione; l'insieme completo degli *use cases* si può analizzare consultando l'appendice B. Tutti gli *use cases* così come i diagrammi riportati nel testo sono scritti in lingua inglese; ciò è giustificato dal fatto che il seguente progetto è stato sviluppato in un contesto internazionale presso la Technische Universität Hamburg-Harburg.

- **Creazione di una nuova annotazione**

  La figura 5.1 rappresenta la funzionalità di creazione di una nuova annotazione; da notare il meccanismo di identificazione dell'utente che realizza la nuova annotazione e che deve quindi essere stato registrato precedentemente nel sistema.

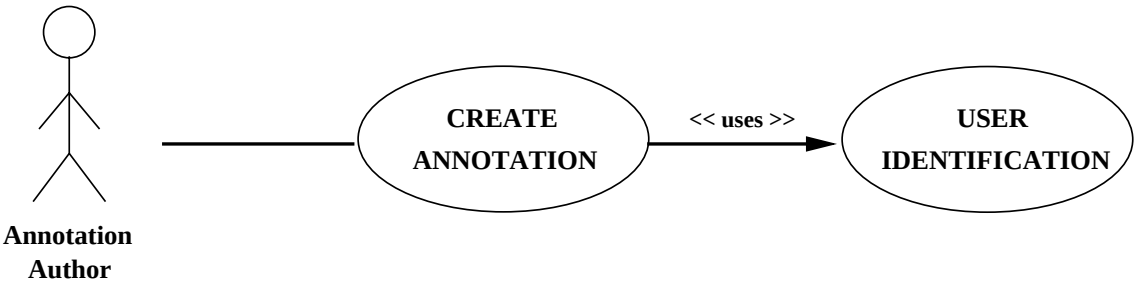

Figura 5.1: Creazione di una nuova annotazione.

La figura 5.2 mostra più in dettaglio i passi attraverso cui un utente effettua la creazione di una nuova annotazione.

Anzitutto viene selezionato il documento o l'annotazione che si desidera annotare; quindi, una volta trovato il documento desiderato, si crea l'annotazione definendone il testo, i soggetti a cui si riferisce e gli *Access Modifiers*. A questo punto l'*Annotation Server* attiva la procedura di notificazione in base alle regole spiegate nella sezione 5.8.2.

[5] Il *Project Administrator* è una figura amministrativa che svolge tutte le funzioni relative alla gestione del gruppo di progetto.

[6] Il *System Administrator* è una figura amministrativa che si occupa della gestione dell'intero *Annotation Server*.

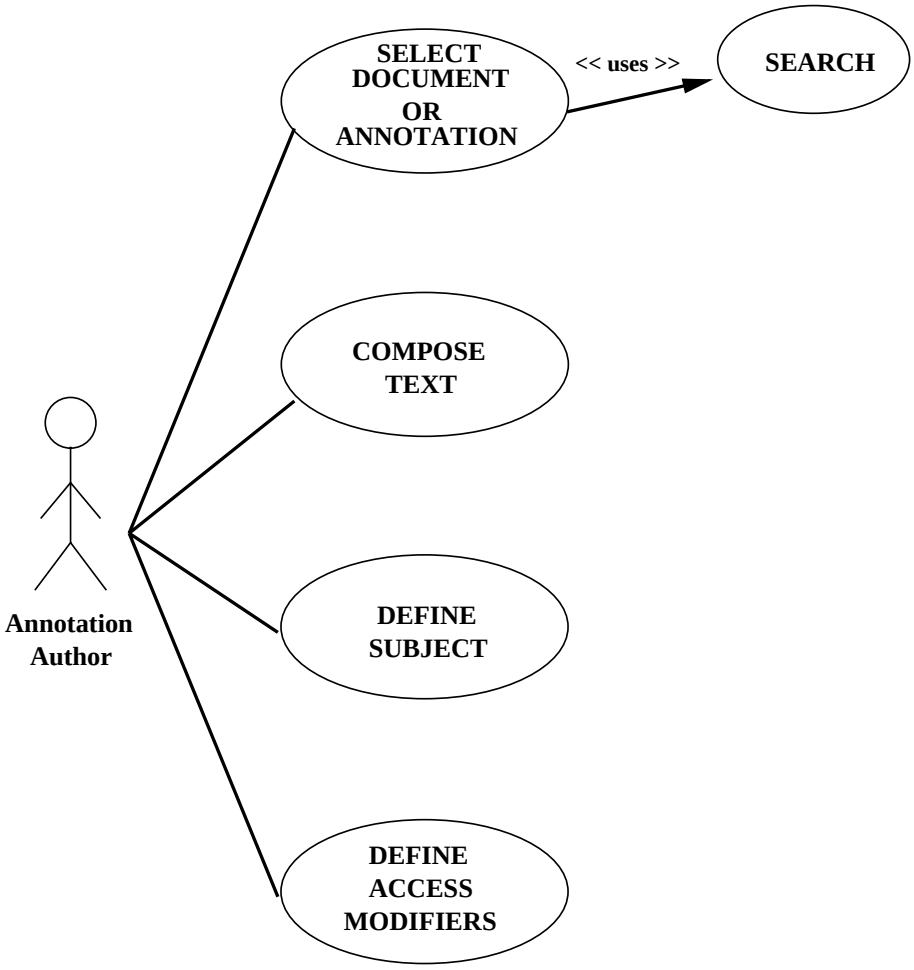

Figura 5.2: Creazione di una nuova annotazione -Dettagli-.

- **Notificazione attraverso *e-mails***

  Il meccanismo di notificazione è realizzato mediante l'utilizzo della posta elettronica; in tal caso si parla anche di notificazione *off-lines*. Si possono individuare tre situazioni in cui occorre effettuare una notifica tramite *e-mail* agli utenti interessati:

  – Notifica a seguito di nuova annotazione
  – Notifica a seguito di cancellazione
  – Notifica a seguito di inserimento o cancellazione di un nuovo utente da un gruppo di progetto

  Ciò significa che il meccanismo di notificazione deve essere attivato ogni volta che un qualsiasi utente crea un'annotazione oppure quando l'amministratore del sistema effettua una cancellazione oppure quando un amministratore di progetto inserisce od elimina un nuovo membro dal gruppo di progetto. Le regole in base alle quali è realizzato l'algoritmo

di notifica sono esposte in maniera esaustiva nella sezione 5.8.2. La figura 5.3 mostra in generale la funzionalità di notificazione realizzata dall'*Annotation Server*.

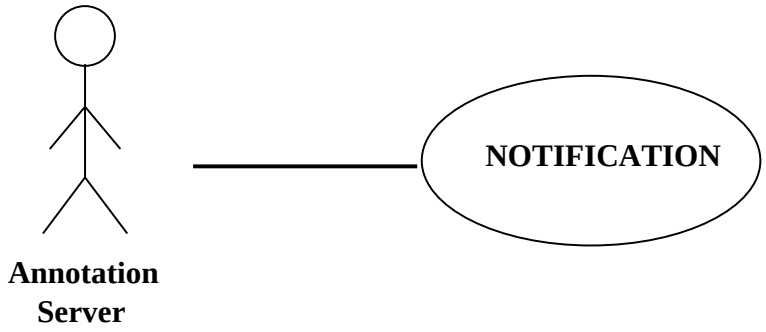

Figura 5.3: Meccanismo di notificazione.

La figura 5.4 mostra in dettaglio la procedura di notificazione. In tale *use case* l'attore è rappresentato dall'*Annotation Server* che è un componente dell'architettura del sistema descritta nella sezione 7.1.

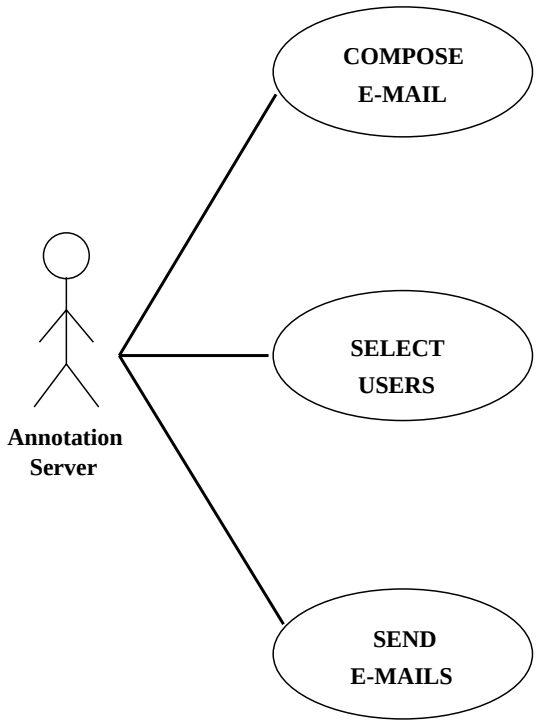

Figura 5.4: Meccanismo di notificazione -Dettagli-.

Ogni volta che viene effettuata un'operazione che richiede la procedura di notificazione (creazione di una nuova annotazione, cancellazione di un'annotazione già esistente oppure inserimento/cancellazione di un'utente da un gruppo di progetto) l'*Annotation Server* in

primo luogo determina quali utenti, registrati nel sistema, dovranno ricevere la notifica. Quindi l'*Annotation Server* compone una *e-mail* da spedire ad ogni utente opportunamente selezionato. L'*e-mail* contiene tutte le informazioni necessarie per comprendere cosa sia accaduto nel sistema.

La notificazione è un aspetto molto importante dell'*Annotation Server* che lo rende un componente attivo, capace cioè di reagire a degli eventi, eseguendo delle azioni nel caso in cui determinate condizioni siano verificate.

- **Cancellazione di un'annotazione**

  L'*Annotation Server* è in grado di attivare un meccanismo di notificazione anche nel momento in cui un'annotazione viene cancellata.

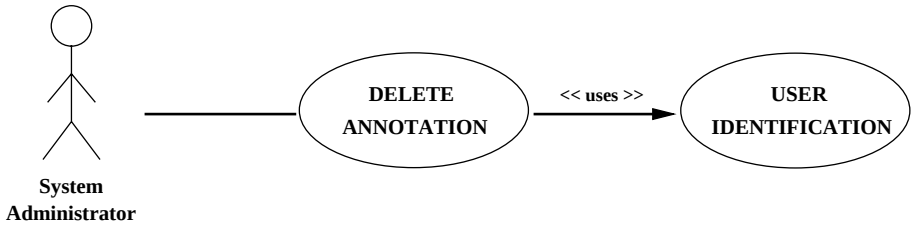

Figura 5.5: Cancellazione di un'annotazione.

Anzitutto, come si può osservare dalla figura 5.5, solo il *System Administrator* può effettuare la cancellazione di un'annotazione.

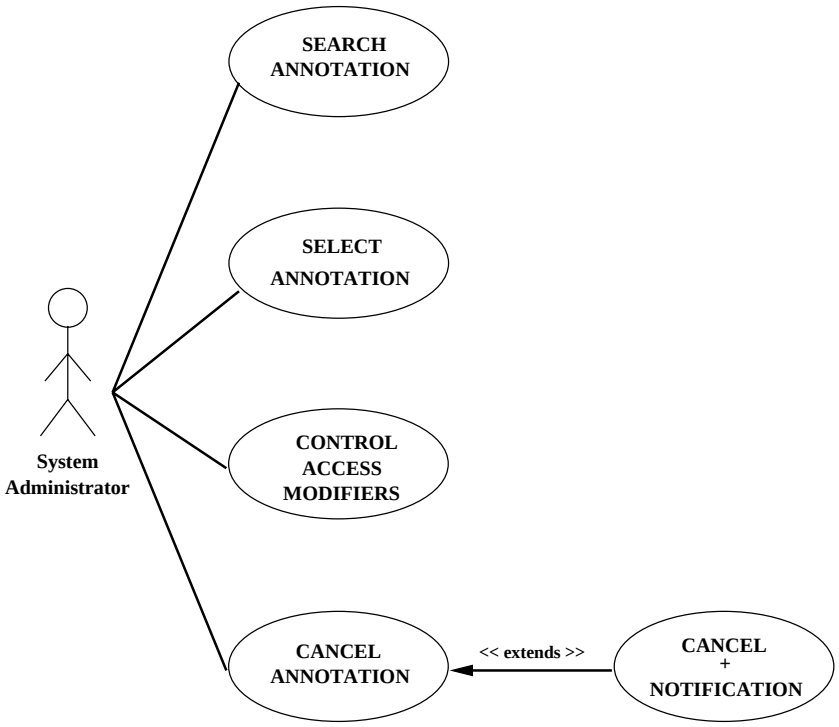

Figura 5.6: Cancellazione di un'annotazione -Dettagli-.

La figura 5.6 mostra in dettaglio i passi che vengono compiuti dal sistema nel momento in cui viene eseguita la procedura di cancellazione. A seguito di cancellazione viene attivato il meccanismo di notificazione in base ai criteri descritti nella sezione 5.8.3. La notificazione avviene, come nel caso di creazione di una nuova annotazione e nel caso di inserimento di un utente in un gruppo di progetto, attraverso la selezione degli utenti interessati e la spedizione di una *e-mail*. La cancellazione di un'annotazione è un'operazione estremamente delicata e che quindi necessita qualche osservazione aggiuntiva. Nel momento in cui un'annotazione viene cancellata si pone il grosso quesito di quale dovrà essere il destino dell'eventuale gerarchia di annotazioni sottostante. Purtroppo una soluzione univoca a tale problema non esiste in quanto le decisioni possibili sono diverse a seconda della semantica delle annotazioni sottostanti. Una descrizione dettagliata del problema relativo alla cancellazione è riportata nella sezione 5.8.3. In questo momento ciò che è necessario puntualizzare è che la cancellazione di annotazioni deve essere un'operazione effettuata con una frequenza molto bassa in quanto determina l'automatica eliminazione di tutte le annotazioni che eventualmente si riferiscono a quella originale. Per tale motivo è stata fatta la scelta di dare la possibilità di cancellare annotazioni solo al *System Administrator* il quale dovrà valutare con cura le situazioni in cui tale operazione è opportuna.

- **Visualizzazione di una serie di annotazioni**

  La figura 5.7 mostra la funzionalità generica di visualizzazione delle annotazioni messa a disposizione dal sistema.

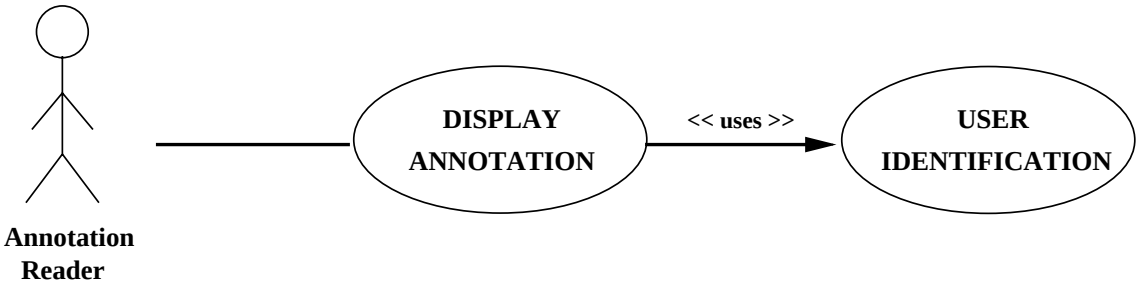

Figura 5.7: Visualizzazione di una serie di annotazioni.

Come è mostrato nella figura 5.7 l'utente che desidera visualizzare le annotazioni in base ad un qualsiasi criterio deve prima identificarsi al sistema che provvederà quindi a determinare i diritti da lui posseduti.

- **Visualizzazione delle annotazioni di un determinato progetto**

  Tra le varie modalità offerte dal sistema l'utente può ricercare tutte le annotazioni che si riferiscono ad un determinato progetto che appartiene alla gerarchia dei progetti esistenti. In tal caso egli seleziona il progetto ed il sistema, dopo opportuni controlli, rende disponibili le annotazioni individuate. Tale funzionalità è riportata nella figura 5.8 e rappresenta in modo dettagliato l'*use case* riportato nella figura 5.7.

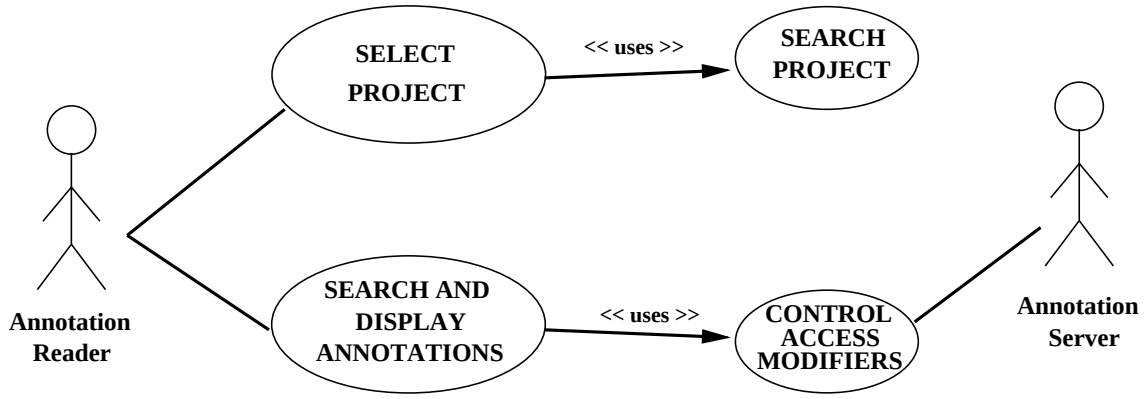

Figura 5.8: Visualizzazione delle annotazioni di un determinato progetto.

- **Visualizzazione delle annotazioni di una determinata persona**

  Il sistema offre anche la possibilità di visualizzare l'insieme delle annotazioni che sono state effettuate da una determinata persona e che l'utente ha il diritto di leggere. Il sistema effettua i controlli necessari e visualizza l'insieme delle annotazioni selezionate (figura 5.9).

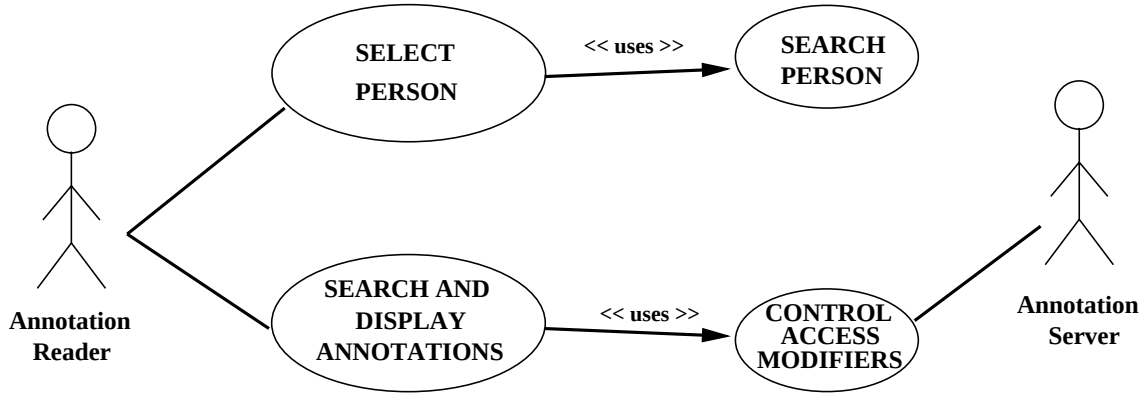

Figura 5.9: Visualizzazione delle annotazioni di una determinata persona.

5.3 Il modello

Scopo di tale sezione è la descrizione dei principi che stanno alla base del modello sviluppato. Molti dei concetti di seguito riportati verranno ripresi in maniera più approfondita nelle sezioni successive. La scelta di tale modalità espositiva è dettata dal fatto che una visione generale dell'intero modello aiuta poi a comprendere più facilmente i dettagli e le motivazioni che hanno portato alla definizione delle varie entità e delle loro relazioni.

La figura 5.10 mostra, utilizzando la notazione prevista dal linguaggio UML, la struttura complessiva del modello. Il primo elemento da prendere in considerazione è l'oggetto *Workspace*; un generico spazio di lavoro può essere, come mostra la gerarchia riportata in figura, un

oggetto *Public* oppure un oggetto *Project*. L'oggetto *Public* non è un vero e proprio progetto ma è semplicemente il nodo radice della gerarchia dei progetti; in altri termini tutti i progetti hanno come *superproject*, indirettamente o direttamente, l'oggetto *Public* (sezione 5.5.3). L'oggetto *Project* invece definisce un vero e proprio progetto che può avere più sottoprogetti e può riferirsi direttamente ad un oggetto *Public*.

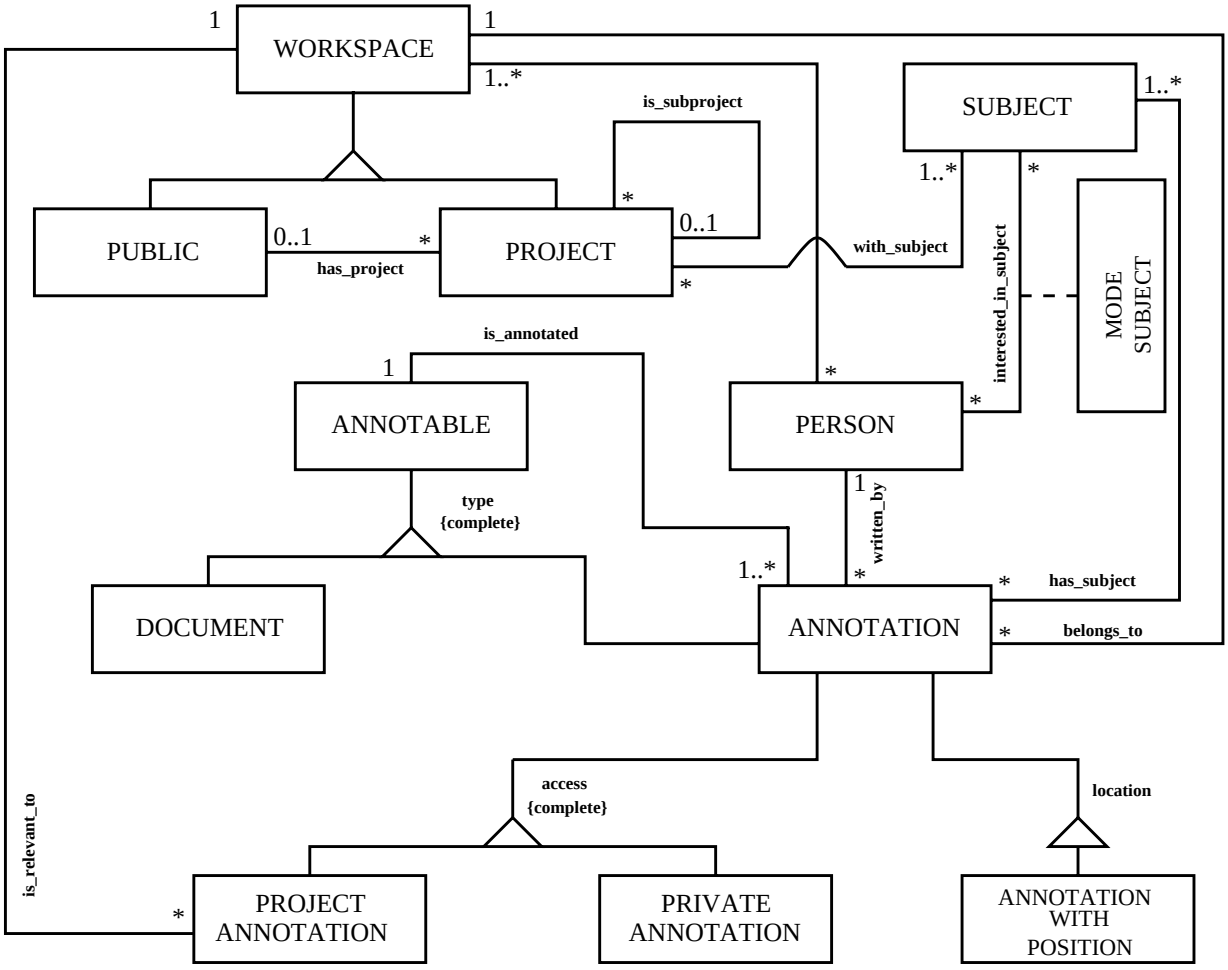

Figura 5.10: Modello di base del sistema.

Altro elemento del modello è il *Subject*; ogni progetto nel momento in cui viene creato deve avere almeno un soggetto[7] altrimenti non avrebbe alcun senso il progetto medesimo. Inoltre i *Subjects* possono appartenere anche ad oggetti *Person* oppure *Annotation*.

Ogni persona registrata nel sistema è un'istanza della classe *Person* ed appartiene automaticamente al *Workspace Public*. Ogni utente inoltre per default è interessato a tutti i soggetti dei progetti a cui appartiene esplicitamente.

[7]I soggetti che appartengono ad un determinato progetto rappresentano i *Project Goals* (obiettivi che devono essere raggiunti dal gruppo di progetto).

Il sistema offre la possibilità all'utente di aggiungere o cancellare soggetti che si riferiscono a lui direttamente; inoltre, al fine della realizzazione del meccanismo di notificazione, l'utente può definire la modalità in base alla quale è interessato ad un determinato soggetto. Ciò avviene per mezzo della classe *Mode* che permette di definire se un soggetto di interesse è *Public* oppure rilevante solo nel contesto del progetto in cui è stato definito; se un soggetto di interesse è *Public* l'utente verrà notificato ogni volta che verrà effettuata una nuova annotazione *Public* che ha quel particolare soggetto appartenente all'insieme dei soggetti di interesse dell'utente. In caso contrario l'utente non riceverà alcuna notifica.

L'oggetto principale dell'intero modello è sicuramente l'*Annotation*. Un annotazione può essere riferita ad un documento HTML oppure ad un altra annotazione. Praticamente l'annotazione consiste in una pagina HTML e quindi può prevedere sia testo, che immagini che *links* ad altri documenti. L'annotazione inoltre può essere riferita ad un intero documento oppure a parti dello stesso; questa è la ragione per cui nel modello compare la gerarchia con il nuovo oggetto *Annotation With Position*.

Le annotazioni inoltre possono essere *Private* oppure effettuate a livello di un determinato progetto. Ogni annotazione per default appartiene al progetto nell'ambito del quale è stata effettuata ma le annotazioni di progetto possono essere rilevanti per un progetto diverso da quello corrente (ad esempio per un progetto che si trova ai livelli più alti della stessa gerarchia). Per una spiegazione più approfondita di tali concetti fare riferimento alla sezione 5.4. Ogni annotazione inoltre possiede per default tutti i soggetti del progetto a cui appartiene; l'utente che crea l'annotazione può modificare tale lista con il vincolo che l'annotazione deve possedere almeno un soggetto. Nell'appendice B è riportato un diagramma che rappresenta la descrizione dell'intero modello con la sintassi UML; in tale figura sono riportati i nomi di tutte le relazioni, le regole di navigabilità ed anche gli attributi ed i metodi principali di ogni oggetto. Nel seguito di tale capitolo è dedicata ad ogni oggetto una sezione in cui vengono spiegati dettagliatamente gli attributi ed i metodi da esso posseduti.

5.4 *Access Modifiers*

Gli *Access Modifiers* sono le modalità in base alle quali un'annotazione può essere effettuata con riferimento ad uno specifico documento e nell'ambito di un determinato progetto. La logica sottostante agli *Access Modifiers* è di fondamentale importanza per la definizione poi del meccanismo di notificazione. Con riferimento alla struttura ad albero dei progetti (sezione 5.5.3) ed alla possibilità di poter avere anche una gerarchia di annotazioni (sezione 5.8.4) sono stati individuati, dal punto di vista concettuale, tre diversi *Access Modifiers*:

- Public
- Project(P)
- Private

Gli *Access Modifiers* vengono definiti al momento della creazione dell'annotazione e non possono essere più modificati in seguito nel corso della sua vita.

5.4.1 *Public*

Quando un'annotazione è definita *Public* ogni utente registrato nel sistema è abilitato alla lettura di tale annotazione indipendentemente dall'insieme di soggetti legati all'annotazione e dal progetto nell'ambito del quale essa è stata scritta. Le annotazioni *Public* sono tali in quanto il proprietario ritiene che siano di generale interesse e quindi le mette a disposizione di chiunque desideri consultarle.

5.4.2 *Project(P)*

Tale *Access Modifier* è più vincolante rispetto a quello descritto sopra; l'aspetto interessante è che tale modalità d'accesso è parametrica. Il parametro "P" rappresenta un qualsiasi progetto della gerarchia dei progetti che si sviluppa verso il *Workspace Public* a partire dal progetto nell'ambito del quale l'annotazione è effettuata; ciò giustifica la relazione *isRelevantTo* presente nel modello[8] (figura 5.10). Con tale *Access Modifier* si vuole specificare nell'ambito di quale progetto l'annotazione creata ha significato; non tutti gli utenti potranno quindi leggere ed annotare tale annotazione bensì solo quelli che appartengono direttamente od indirettamente (in quanto si trovano nella gerarchia sottostante) al progetto "P". Questo concetto ha una grossa importanza al fine del principio di cooperazione; attraverso annotazioni di tipo *Project(P)* è possibile realizzare un lavoro cooperativo tra le persone che appartengono al progetto "P" ed ai suoi sottoprogetti.

5.4.3 *Private*

Questa è la modalità più restrittiva tra quelle proposte; quando un'annotazione è *Private* solo il proprietario la può gestire, leggere ed eventualmente ulteriormente annotare. Tale concetto è di estrema importanza e costituisce lo strumento offerto all'utente per realizzare documenti personalizzati (capitolo 4).

5.5 Analisi delle classi

In tale sezione verranno descritte brevemente le cinque principali classi[9] che fanno parte del modello definito a supporto del meccanismo di annotazione di documenti multimediali. Per ognuna delle classi verranno identificati gli aspetti caratteristici e le loro relazioni.

5.5.1 *Subject*

La classe *Subject* è molto semplice e modellizza principalmente un tema generico rappresentato da un nome ed a cui possono fare riferimento progetti, annotazioni od utenti. Ciò significa che le

[8] In altri termini un'annotazione può essere scritta nell'ambito di un determinato progetto ma essere specificata rilevante nell'ambito di uno dei progetti che si trovano nella gerarchia soprastante a quello di appartenenza.

[9] Le cinque classi principali del modello sono le seguenti: *Subject*, *Person*, *Workspace*, *Annotable* ed *Annotation*.

annotazioni od i progetti possono avere dei temi a cui si riferiscono in funzione dei loro contenuti e del loro ambito di applicabilità. La figura 5.11 schematizza la classe *Subject*.

Un aspetto interessante riguarda i soggetti a cui fanno riferimento gli utenti; in tal caso ogni persona registrata al sistema di annotazione può fare riferimento ad uno o più soggetti in base ad una modalità che può essere *Public* oppure *NoPublic*. Questa distinzione è rilevante al fine del meccanismo di notificazione a seguito di nuova annotazione. Più specificatamente nel momento in cui viene creata una nuova *Public Annotation* tutti gli utenti interessati in modalità *Public* ad almeno uno dei soggetti a cui si riferisce l'annotazione verranno notificati attraverso una *email*.

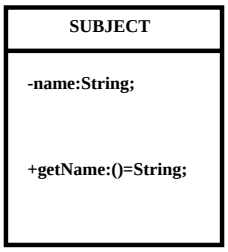

Figura 5.11: Classe *Subject*.

Se vi sono utenti interessati ai soggetti a cui si riferisce l'annotazione pubblica ma in modalità *NoPublic* essi non riceveranno alcuna notifica anche se avranno ovviamente tutti i diritti di lettura su tale nuova annotazione. Tale meccanismo è stato pensato al fine di dare all'utente la possibilità di limitare il numero di *emails* ricevute in quanto, nel caso in cui l'utente abbia molti soggetti di interesse, potrebbe essere fastidioso ricevere una quantità enorme di notifiche.

5.5.2 *Person*

La classe *Person* modellizza un qualsiasi utente registrato nella base di dati; la figura 5.12 riporta lo schema di tale classe con la specifica degli attributi che dovranno essere implementati.

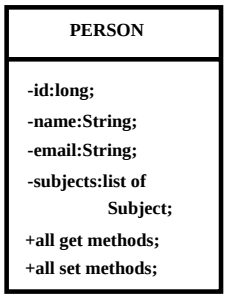

Figura 5.12: Classe *Person*.

Dal punto di vista dell'amministrazione dell'intero sistema e di ogni progetto occorre definire la presenza di due ruoli particolari che sono comunque istanze della classe *Person*: il *System Administrator* ed il *Project Administrator*.

Il *System Administrator* è colui che amministra l'intero sistema ed è l'unico utente che ha il potere di cancellazione delle annotazioni. L'amministratore del sistema non coincide necessariamente con una persona fisica ma, nel modello proposto, è sempre un oggetto della classe *Person* e ne possiede quindi tutti gli attributi specifici. Il *System Administrator* viene creato automaticamente nel momento in cui l'*Annotation Server* viene avviato ed è anche l'amministratore del progetto *Public*. I suoi attributi possono essere ovviamente modificati utilizzando i metodi della classe *Person* descritti nel capitolo 7, ma la sua istanza non può essere cancellata da nessuno e nemmeno dall'amministratore stesso.

Altra figura particolare è quella del *Project Administrator*; anche l'amministratore di progetto non coincide necessariamente con una persona fisica ma rappresenta virtualmente un'istanza della classe *Person*. Ogni progetto deve avere il suo amministratore che è colui che svolge le operazioni di gestione dell'intero progetto quali l'aggiunta o la cancellazione di utenti dal gruppo di lavoro. Altri attori che rientrano nell'insieme degli oggetti della classe *Person* sono l'*Annotation Author* e l'*Annotation Reader* i quali rappresentano rispettivamente generici utenti che creano oppure leggono annotazioni.

In fine ogni generico utente registrato nella base di dati è un oggetto della classe *Person* e ne possiede gli attributi ed i metodi di gestione specificati nella figura 5.12.

5.5.3 *Workspace*

La classe *Workspace* rappresenta un generico spazio di lavoro nell'ambito del quale un determinato gruppo di utenti lavora al fine del conseguimento di determinati obiettivi comuni. La figura 5.13 mostra gli attributi che tale classe deve possedere e la gerarchia ad essa sottostante.

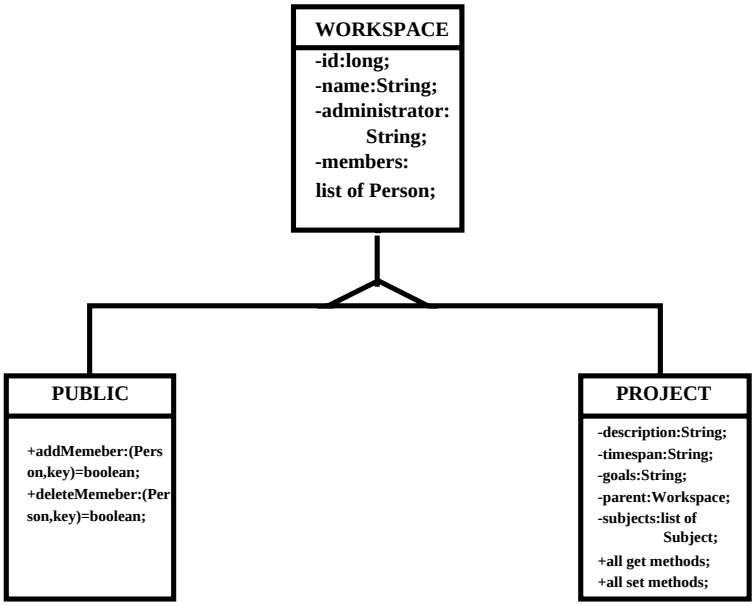

Figura 5.13: Classe *Workspace*.

Come si può osservare dalla figura 5.13 un generico spazio di lavoro può essere un ogget-

to della classe *Public* oppure un oggetto della classe *Project*; la gerarchia è totale ed esclusiva. L'oggetto della classe *Public* rappresenta uno spazio di lavoro virtuale e costituisce il nodo radice nell'albero gerarchico dei progetti; ciò significa che ogni progetto costruito nell'ambito del sistema proposto sarà figlio del *Workspace Public*. Nel momento in cui viene attivato l'*Annotation Server* viene creato automaticamente il *Workspace Public* a cui appartiene il *System Administrator*. L'amministratore del sistema inoltre è anche il *Project Administrator* del *Workspace Public*. Quando un utente viene registrato nel sistema egli appartiene automaticamente al *Workspace Public* nell'ambito del quale potrà iniziare a lavorare ad esempio definendo un proprio progetto, creando delle annotazioni e così via.

La figura 5.14 mostra un esempio di come potrebbe essere strutturata una gerarchia di progetti; come si può notare il *Workspace Public* si trova in cima a tale albero ed ogni progetto, direttamente od indirettamente, si riferisce ad esso.

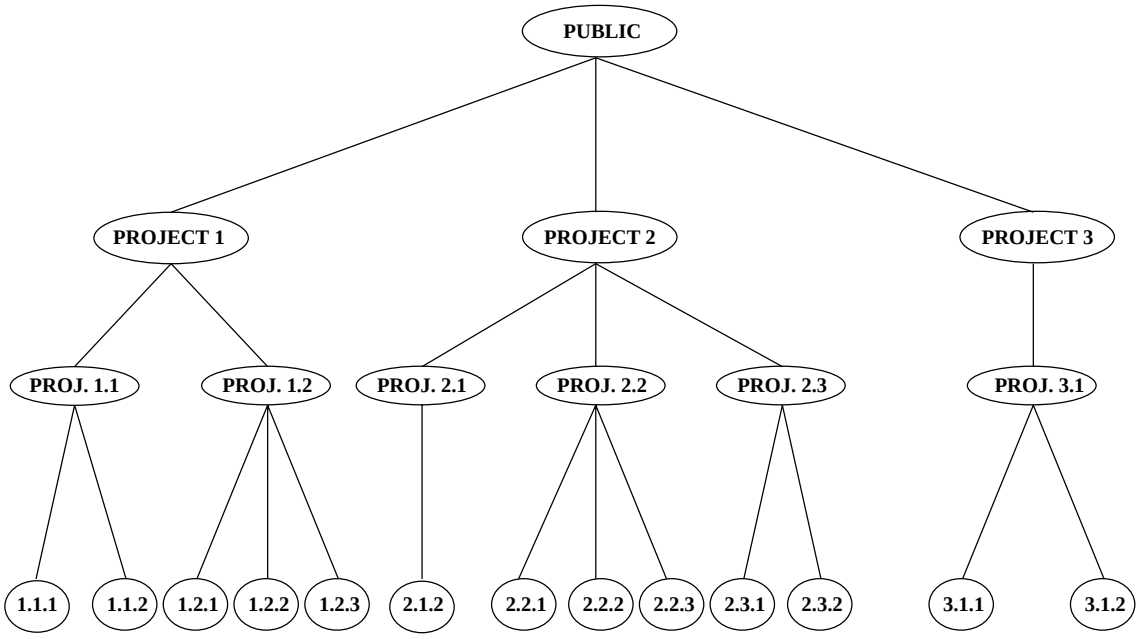

Figura 5.14: Struttura ad albero dei progetti.

La figura 5.15 mostra invece un esempio concreto di un ramo dell'albero gerarchico dei progetti facendo riferimento al progetto WEL (*Warburg Electronic Library*, sezione 4.2.2) nell'ambito del quale è stato sviluppato il prototipo presentato in tale testo.

Altro aspetto importante riguardante l'analisi della classe *Workspace* è relativo alle modalità in base alle quali vengono gestite le annotazioni nell'ambito dei progetti in cui sono realizzate. Ogni annotazione, sia essa *Private*, *Project(P)* oppure *Public*, ha un progetto a cui appartiene che è quello nell'ambito del quale lavora l'utente nel momento in cui crea l'annotazione. Le annotazioni *Project(P)* inoltre possono essere definite rilevanti per un progetto diverso rispetto a quello di appartenenza; ciò significa che tali annotazioni avranno un progetto nell'ambito del quale sono state effettuate ed un progetto per cui sono rilevanti ovvero nell'ambito del quale possono essere lette ed eventualmente annotate. Tale concetto è molto importante in quanto rende l'intero

sistema notevolmente flessibile realizzando il principio di collaborazione a lungo discusso in tale testo.

È inoltre importante specificare l'importanza della gerarchia dei progetti per quanto riguarda i membri che costituiscono i gruppi di lavoro; ogni utente che fa parte di un gruppo di progetto indirettamente è anche membro dei progetti che si trovano nella gerarchia soprastante. Questo concetto è di fondamentale importanza in quanto influenza enormemente i principi in base ai quali sono organizzate le annotazioni e che sono esposti in modo esaustivo nel corso di tale capitolo.

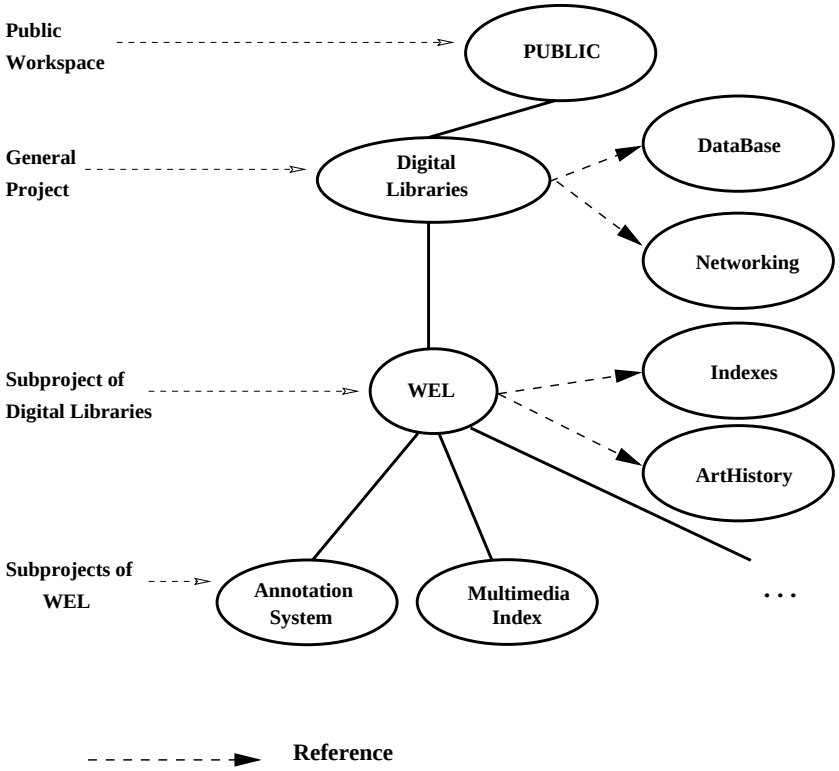

Figura 5.15: Esempio di una gerarchia di progetti.

Ultimo aspetto che merita qualche considerazione è quello relativo al nome del progetto; nel momento in cui un utente crea un progetto egli deve anzitutto avere fatto un *login* con il sistema e quindi deve già lavorare nell'ambito di un determinato progetto che al più può essere il *Workspace Public*. Il nuovo progetto viene creato automaticamente come sottoprogetto di quello nell'ambito del quale sta lavorando l'utente; l'utente inoltre specifica il nome del nuovo progetto ed il sistema crea automaticamente un nome completo che identifica il progetto all'interno della gerarchia utilizzando il meccanismo della *Dot Notation*. Ogni progetto così come ogni utente ed ogni soggetto deve avere un nome univoco nell'ambito dell'intera base di dati.

Al fine di chiarire i concetti esposti in tale sezione è bene fare un esempio di un possibile scenario facendo riferimento alla gerarchia riportata nella figura 5.15. Un utente appartenente al progetto *Annotation System* potrebbe fare delle *Private Annotations* riferite a documenti od ad

altre annotazioni; in tal caso nessun altro utente avrà diritti di lettura su tali annotazioni. Inoltre egli potrà fare anche delle *Project Annotations* che per definizione apparterranno al progetto *Annotation System* nell'ambito del quale sta lavorando l'utente, ma che potranno essere definite rilevanti per i progetti *WEL, Digital Libraries* oppure *Public*. Da notare che un utente che lavora nell'ambito del progetto *Annotation System* non può definire annotazioni rilevanti per il progetto *Multimedia Index* o per eventuali sottoprogetti del progetto *Annotation System* ma solo per progetti che si trovano nella gerarchia ad esso soprastante.

5.5.4 *Annotable*

La classe *Annotable* rappresenta un generico elemento che può essere oggetto di annotazione. Come mostra la gerarchia riportata in figura 5.16 ci possono essere due tipologie di oggetti che vengono annotati: *Document* ed *Annotation*.

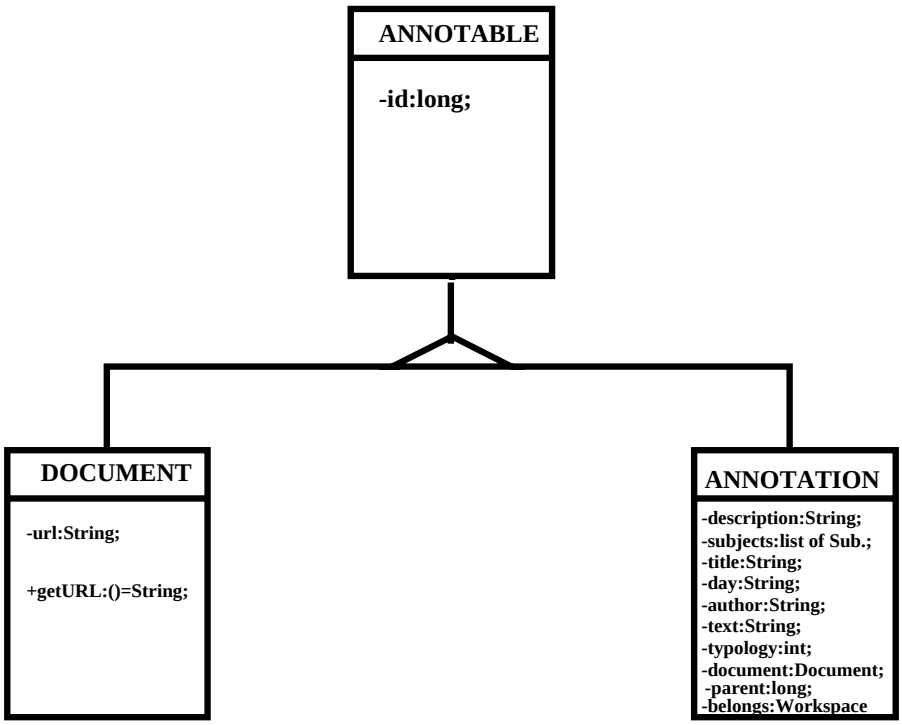

Figura 5.16: Classe *Annotable*.

Ogni oggetto della classe *Document* rappresenta, per mezzo del suo URL, un generico documento presente sul WWW. Nella base di dati viene registrato solamente il riferimento al documento e non l'intera pagina HTML; sarà poi il *Proxy Server* (fare riferimento all'architettura del sistema riportata nella sezione 7.1) ad occuparsi del reperimento del documento e della fusione con le annotazioni che ad esso si riferiscono.

La classe *Annotable* è stata introdotta in quanto, a livello concettuale, le annotazioni sono anch'esse dei documenti annotabili, ovvero possono essere oggetto di annotazione come i

normali documenti residenti sul WWW. Attraverso l'introduzione della gerarchia di figura 5.16 è possibile modellizzare il fatto che l'annotazione può essere riferita ad un documento oppure ad un'altra annotazione, cosa questa che non sarebbe possibile se le due classi *Annotation* e *Document* fossero lasciate separate.

5.5.5 *Annotation*

La classe *Annotation* è l'elemento centrale di tutto il sistema; la figura 5.17 mostra in modo specifico quali sono gli attributi di tale classe.

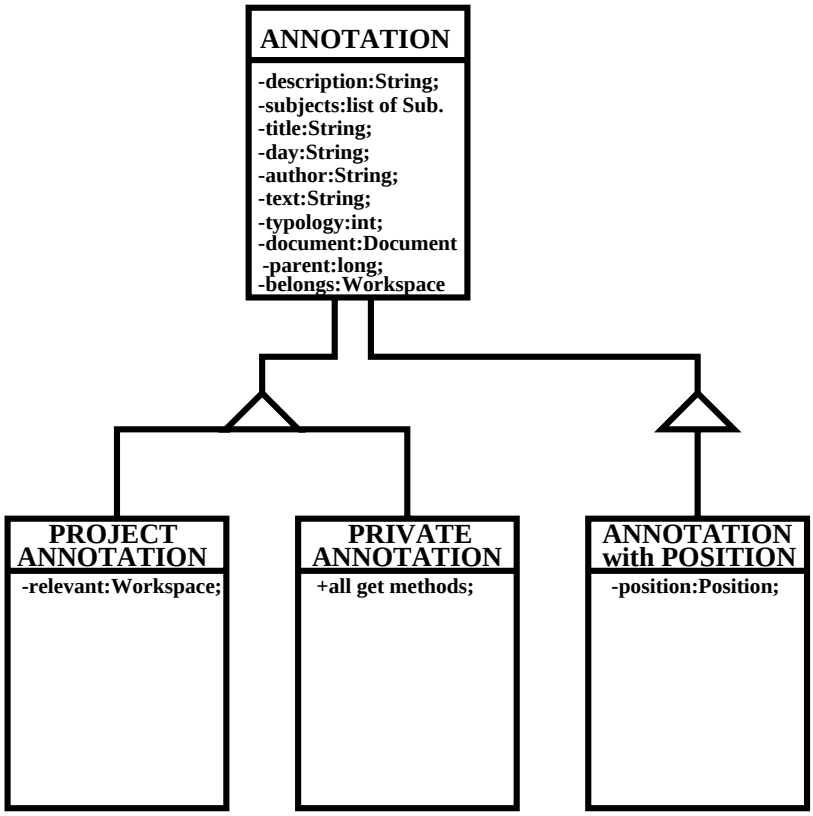

Figura 5.17: Classe *Annotation*.

Come si può osservare dal modello del sistema riportato in figura 5.10 al di sotto della classe *Annotation* si strutturano due gerarchie che sono tra di loro ortogonali. Anzitutto le annotazioni possono essere *Project* oppure *Private*; le annotazioni *Private* possono essere lette ed annotate solamente dal loro autore e consentono la realizzazione di strutture personalizzate. Le annotazioni *Project* sono fatte nell'ambito di un determinato progetto e possono essere specificate rilevanti per un progetto che si trova nella gerarchia soprastante a quello di appartenenza. Tale meccanismo consente la realizzazione di un lavoro collaborativo tra membri appartenenti a progetti diversi. In fine le annotazioni possono essere riferite all'intero oggetto annotabile (documento od annotazione) oppure a sue parti interne quali porzioni di testo od immagini in esso contenute.

Ogni annotazione deve avere inoltre un autore ed almeno un soggetto a cui si riferisce il quale viene specificato in fase di creazione.

Per avere una visione complessiva dell'intero sistema fare riferimento allo schema completo del modello riportato in figura B.29 nell'appendice B.

5.6 Progetto della base di dati

Oggetto di tale sezione è il progetto della base di dati gestita dall'*Annotation Server* (fare riferimento all'architettura del sistema descritta nella sezione 7.1). Di seguito sono riportati gli aspetti generali riguardanti le entità individuate ed i cammini di join più significativi; una descrizione più dettagliata con la specifica del tipo di ogni attributo per ogni entità è riportata nel capitolo 7 e nell'appendice B.

La metodologia che si è seguita in fase di progettazione della base di dati è riportata schematicamente nella figura 5.18; per avere informazioni circa la semantica e la sintassi dei modelli e degli schemi utilizzati fare riferimento a [S.Ceri *et al.* 1996].

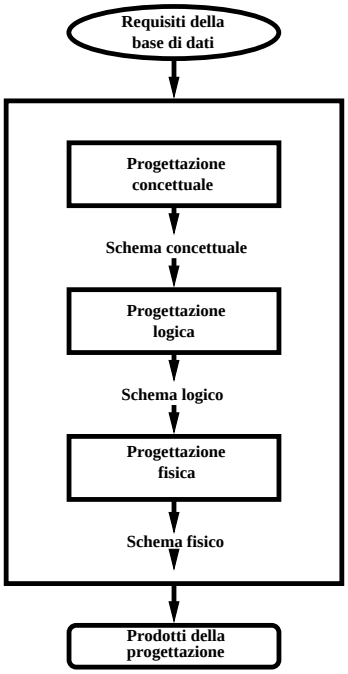

Figura 5.18: Fasi di progettazione della base di dati.

5.6.1 Progettazione concettuale

Scopo della progettazione concettuale è la rappresentazione delle specifiche informali della realtà di interesse in termini di una descrizione formale e completa, ma indipendente dai criteri di rappresetazione utilizzati nei sistemi di gestione di basi di dati [S.Ceri *et al.* 1996]. Il prodotto di questa fase viene chiamato schema concettuale e fa riferimento ad un modello concettuale dei

dati attraverso cui il progettista rappresenta il contenuto informativo della base di dati senza fare riferimento a specifici aspetti implementativi. Nella figura 5.19 è riportato lo schema concettuale della base di dati a supporto del sistema di annotazione proposto in tale testo. Tale schema è realizzato per mezzo del modello Entità-Relazione la cui sintassi è riportata in modo specifico in [S.Ceri *et al.* 1996]; la decisione di utilizzare il modello Entità-Relazioni è giustificata dal fatto che in fase di implementazione si utilizzerà una base di dati relazionale. In tal caso quindi il modello ER è particolarmente appropriato e permette di modellizzare facilmente e con precisione tutti gli aspetti che dovranno poi essere implementati in fase di progettazione fisica.

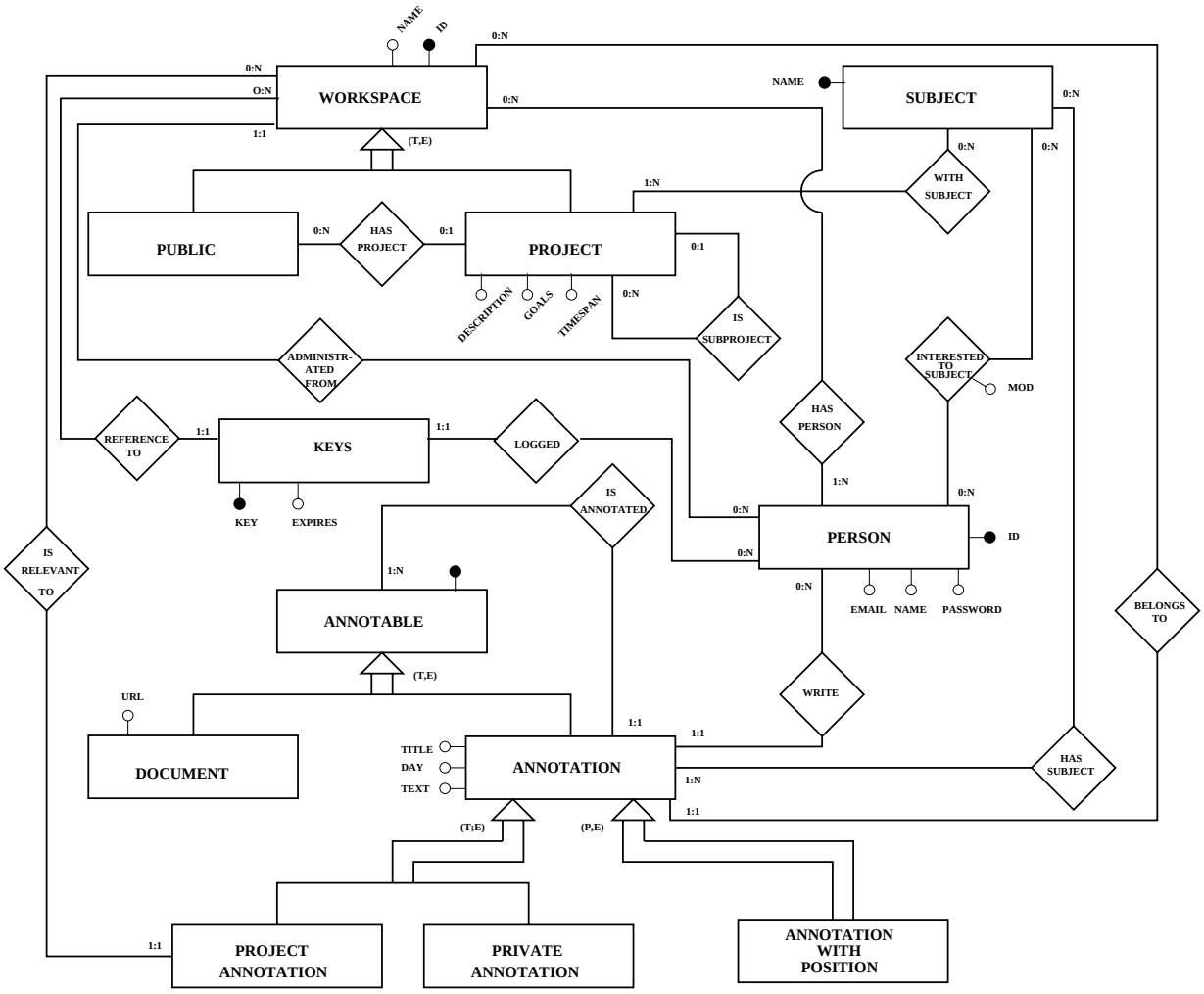

Figura 5.19: Schema concettuale della base di dati.

Lo schema riportato in figura 5.19 è stato realizzato al fine della progettazione della base di dati e presenta delle caratteristiche che lo differenziano dallo schema riportato nella figura 5.10 della sezione 5.3 e dallo schema riportato nella figura B.29 dell'appendice B. Nella figura 5.19 è presente l'entità *KEYS* e l'entità *PERSON* possiede anche l'attributo *Password*. Questo perché

l'*Annotation Server* si deve preoccupare di gestire anche la sicurezza degli accessi attraverso il meccanismo di *login* spiegato nella sezione 7.2 e che quindi non deve essere formalizzato negli schemi concettuali che rappresentano il modello bensì nello schema concettuale che rappresenta la base di dati.

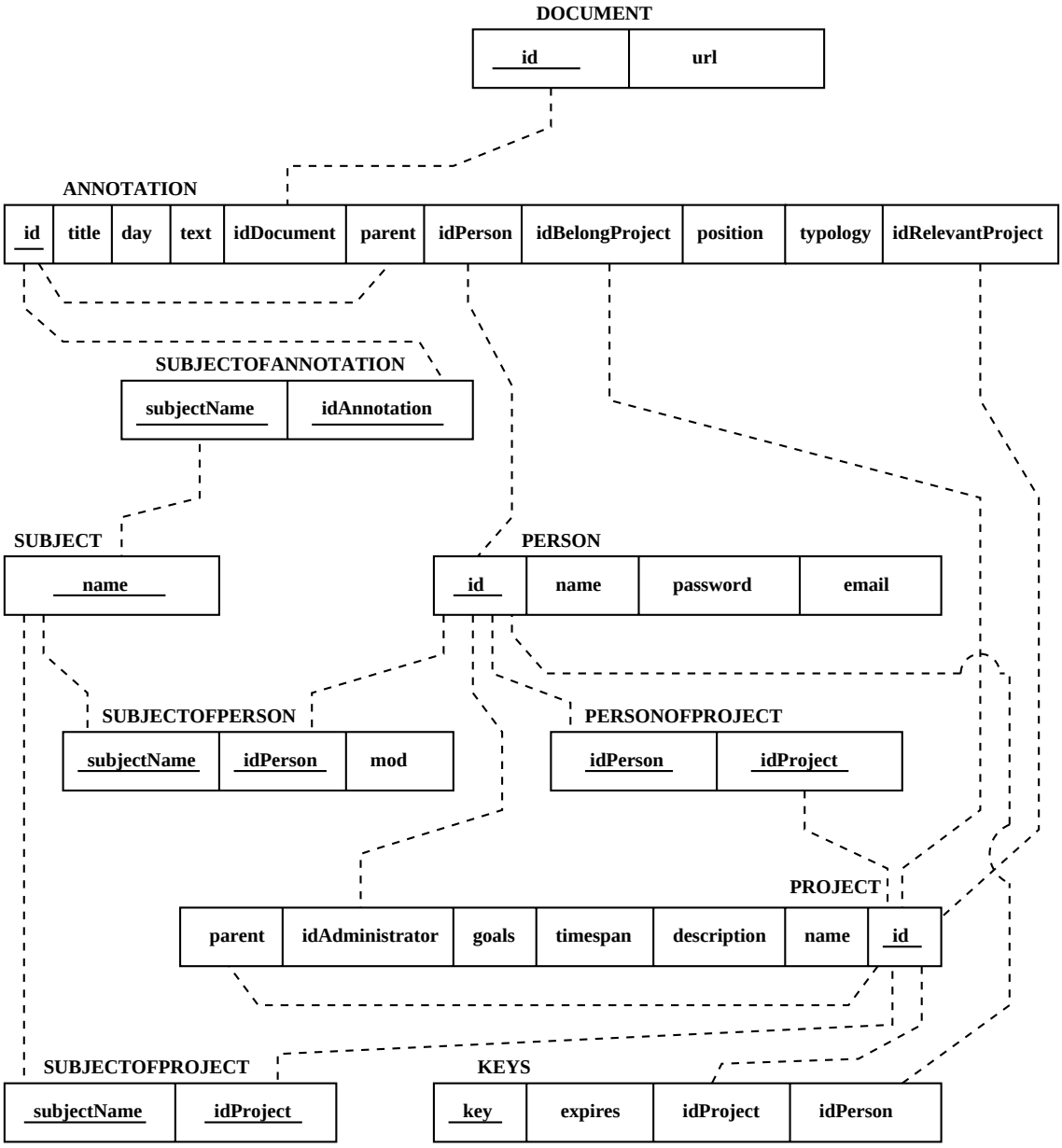

Figura 5.20: Schema logico della base di dati.

Nel diagramma ER di figura 5.19 sono indicate esplicitamente le chiavi, ovvero gli attributi che identificano univocamente ciascuna tupla delle tabelle modellizzate; questo è un aspetto strettamente caratteristico delle basi di dati relazionali.

L'entità KEYS presenta l'attributo *expires* che contiene la data in cui è stato effettuato il *login*

identificato dalla specifica tupla della tabella in questione. Tale attributo è di estrema importanza in quanto sull'*Annotation Server* è realizzato anche un meccanismo di *Garbage Collection*, discusso nella sezione 7.2, che fa uso dell'attributo *expires* per determinare quali righe della tabella KEYS possono essere cancellate in quanto ormai prive di significato.

Ciascuna tupla della tabella KEYS rappresenta quindi un *login* effettuato dall'utente; al momento del *login* il sistema determina in modo casuale una chiave che viene associata all'utente e che dovrà essere utilizzata ogni volta che si richiede lo svolgimento di un particolare servizio da parte dell'*Annotation Server*. La tabella KEYS sta quindi alla base del meccanismo di autentificazione realizzato sull'*Annotation Server*.

5.6.2 Progettazione logica

La progettazione logica consiste nella traduzione dello schema concettuale riportato nella sezione precedente (figura 5.19), in termini di strutture di rappresentazione proprie del tipo di sistema di gestione della base di dati a disposizione.

Il prodotto di tale fase viene denominato schema logico della base di dati e fa riferimento ad un modello logico dei dati. Nel prototipo proposto si è utilizzata una base di dati relazionale ed in figura 5.20 è riportato lo schema logico elaborato con l'indicazione dei più importanti cammini di join. Le regole di traduzione di uno schema concettuale in uno schema logico sono esaurientemente riportate in [S.Ceri *et al.* 1996].

5.7 Dinamica del sistema

La descrizione degli aspetti dinamici del sistema è sempre una questione molto delicata ed a volte estremamente complessa per poter essere esposta con sufficiente chiarezza. Il linguaggio UML a tal proposito fornisce tre strumenti per la definizione della dinamica del sistema noti come *Interaction Diagrams*, *State Diagrams* ed *Activity Diagrams* (una breve descrizione di tali meccanismi è riportata nel capitolo 6). In fase di progettazione del sistema proposto si è deciso di utilizzare in modo piuttosto limitato tali strumenti e con riferimento solamente ad aspetti particolarmente significativi del modello. Nelle sezioni successive verranno riportati alcuni esempi di *Activity Diagrams* allo scopo di definire flussi di lavoro attraverso la descrizione di processi paralleli. Gli *Activity Diagrams* sono particolarmente utili per la descrizione di programmi concorrenti ed il concetto che sta alla loro base è quello di attività; un'attività è un generico compito che può essere svolto da un essere umano oppure da un calcolatore. Ogni attività può essere seguita da un'altra attività in una semplice sequenza; inoltre più attività possono essere concorrenti, ovvero eseguite contemporaneamente in base ad un modello di esecuzione parallela dei compiti.

5.8 Strategie per la gestione delle annotazioni

Le strategie adottate per la gestione delle annotazioni rappresentano il fulcro del prototipo proposto. Per strategia si intende, in tale contesto, la modalità in base alla quale è stata organizzata

la soluzione ai problemi sollevati nelle fasi di modellizzazione precedentemente descritte. La tabella 5.1 riassume sinteticamente le quattro funzionalità più critiche offerte dal sistema:

- Lettura di un'annotazione
- Meccanismo di annotazione ricorsiva
- Notificazione a seguito di nuova annotazione
- Notificazione a seguito di cancellazione

Queste operazioni sono analizzate in correlazione con gli *Access Modifiers*, ovvero con le modalità in base alle quali vengono create le annotazioni. Tale analisi è di estrema importanza in quanto ogni operazione sopra elencata assume caratteristiche diverse a seconda delle modalità in base alle quali è stata scritta una determinata annotazione.

FUNZIONALITA' / ACCESS MODIFIERS	Lettura	Access Modifiers per annotazioni ricorsive	Notificazione a seguito di nuova annotazione	Notificazione a seguito di cancellazione
PUBLIC (A e' un' annotazione appartenente al Workspace Public)	L'annotazione A puo' essere letta da tutti gli utenti registrati nella base di dati.	Sia A' un'annotazione riferita ad A. A' puo' essere: PUBLIC oppure PROJECT(P) oppure PRIVATE	Vengono notificate solo le persone interessate ad almeno uno dei soggetti a cui si riferisce l'annotazione A i quali hanno espresso la volonta' di essere notificati.	Vengono notificati gli autori delle annotazioni presenti nella gerarchia sottostante ad A. Ciascuno riceve una copia dell'annotazione cancellata e l'indicazione dell'elemento a cui essa si riferiva.
PROJECT(P) (A e' un'annotazione rilevante nell'ambito del progetto P specificato come parametro)	L'annotazione A puo' essere letta da tutti i membri del progetto P (inclusi quindi anche tutti i membri dei sottoprogetti del progetto P).	Sia A' un'annotazione riferita ad A. A' puo' essere: PROJECT(P) oppure PROJECT(P') oppure PRIVATE dove P' e' un qualsiasi sottoprogetto di P.	Vengono notificati i membri del progetto P che sono interessati ad almeno uno dei temi dell'annotazione A.	Vengono notificati gli autori delle annotazioni presenti nella gerarchia sottostante ad A. Ciascuno riceve una copia dell'annotazione cancellata e l'indicazione dell'elemento a cui essa si riferiva.
PRIVATE (A e' un'annotazione privata, ovvero appartenente al suo autore)	L'annotazione A puo' essere letta solamente dal suo autore.	Sia A' un'annotazione riferita ad A. A' puo' essere: PRIVATE dello stesso proprietario di A.	Non vi e' alcuna notificazione.	Viene effettuato una notifica all'autore nel caso in cui vi sono altre sue annotazioni nella gerarchia sottostante e quindi si procede alla cancellazione.

Tabella 5.1: Strategie per la gestione delle annotazioni.

La tabella 5.1 riporta ordinatamente tali apsetti mettendo in evidenza le diverse logiche che stanno alla base delle quattro operazioni (lettura, annotazione ricorsiva, notificazione a seguito di nuova annotazione, notificazione a seguito di cancellazione) in funzione degli *Access Modifiers* (*Public, Project(P)* oppure *Private*) delle annotazioni.

Le colonne più interessanti di tale tabella sono le ultime due posizionate a destra, le quali riportano sinteticamente le regole che stanno alla base del meccanismo di notificazione.

I principi riportati e le strategie adottate non rappresentano sicuramente l'unica soluzione possibile ai problemi analizzati; esse rappresentano la soluzione che è stata ritenuta migliore in dipendenza delle strutture gerarchiche dei progetti e delle annotazioni che si è deciso di adottare e che sono esposte nel corso di tale capitolo.

Il prototipo proposto inoltre si inserisce nell'ambito di un progetto di lungo termine e quindi potrà prevedere sicuramente successive versioni che implementeranno nuove soluzioni ed aggiungeranno nuove funzionalità (capitolo 9).

Le sezioni successive analizzano dettagliatamente le funzionalità offerte dal sistema in dipendenza degli *Access Modifiers* alle annotazioni, prendendo in considerazione tutte le possibili combinazioni che si possono presentare.

5.8.1 Creazione

La creazione di una nuova annotazione è una delle prime funzionalità (tra queste vi è anche la lettura delle annotazioni già esistenti) a cui accede generalmente un nuovo utente dopo essere stato registrato nel sistema. Al momento della registrazione un utente inoltre può essere assegnato ad uno o più specifici progetti oppure essere identificato come utente generico ovvero appartenente al *Workspace Public*. La figura 5.21 riporta l'*Activity Diagram* che rappresenta l'attività di creazione di una nuova annotazione.

Concretamente le annotazioni che un utente può realizzare sono di due tipi:

- *Project Annotation*
- *Private Annotation*

Una *Private Annotation* è molto restrittiva in quanto può essere gestita e quindi consultata solamente dal proprietario ovvero da colui che l'ha effettuata.

Una *Project Annotation* invece viene creata a livello di progetto; tale concetto necessita qualche ulteriore spiegazione. Ogni annotazione, comprese quelle *Private*, viene creata sempre nell'ambito di un determinato progetto a cui appartiene l'autore. Le *Project Annotations* inoltre possono essere definite "rilevanti" per un progetto che può essere identico a quello di appartenenza oppure diverso. Ciò da la possibilità all'utente di definire delle regole di visibilità per la propria annotazione. Infatti una *Project Annotation* fatta a livello di progetto "P" può essere letta da tutte le persone che appartengono ai progetti che si trovano nella gerarchia sottostante a "P".

Una *Public Annotation* non è altro che un caso particolare di *Project Annotation* in cui il progetto "P" è il *Workspace Public*. In tale ultimo caso tutte le persone registrate nel sistema, in quanto appartenenti direttamente od indirettamente al *Workspace Public*, potranno leggere la nuova annotazione.

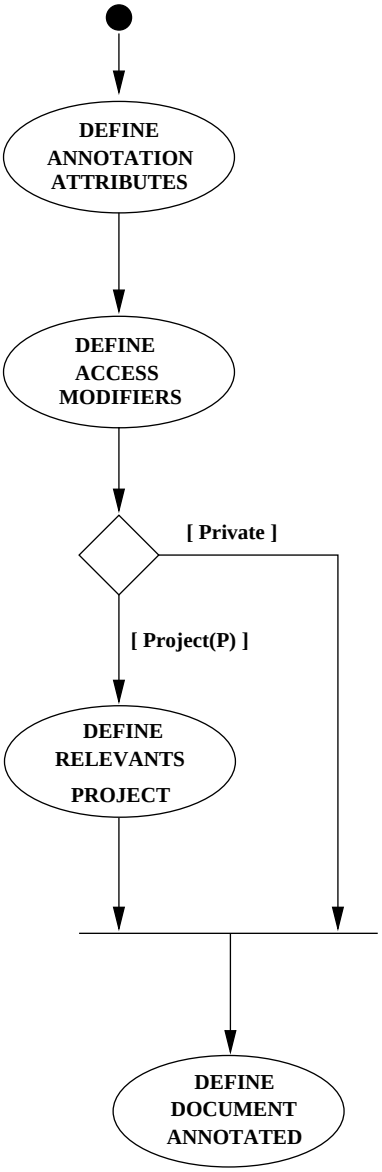

Figura 5.21: Esempio di *Activity Diagram*-Creazione di una nuova annotazione.

La figura 5.22 rappresenta un esempio di come un utente può creare una nuova annotazione utilizzando il prototipo presentato. In tale esempio (figura 5.22) l'utente è membro di due progetti chiamati rispettivamente *WEL* e *Digital Libraries* (Il progetto *Digital Libraries* costituisce un nodo ai livelli gerarchici superiori rispetto al progetto *WEL*[10]); nell'angolo in basso a destra è rappresentato schematicamente l'*Annotation Server* che si occupa di gestire la base di dati in cui vengono memorizzate le annotazioni. L'utente effettua una *Project Annotation* riferita ad un determinato documento nell'ambito del progetto *WEL*; tale annotazione però è definita rilevante per i membri del progetto *Digital Libraries*. Ciò significa che tutti i membri del progetto *Digital*

[10]Fare riferimento alla gerarchia riportata nella figura 5.15.

Libraries, compresi quelli dei suoi sottoprogetti (e quindi anche i membri del progetto *WEL*), potranno leggere tale annotazione.

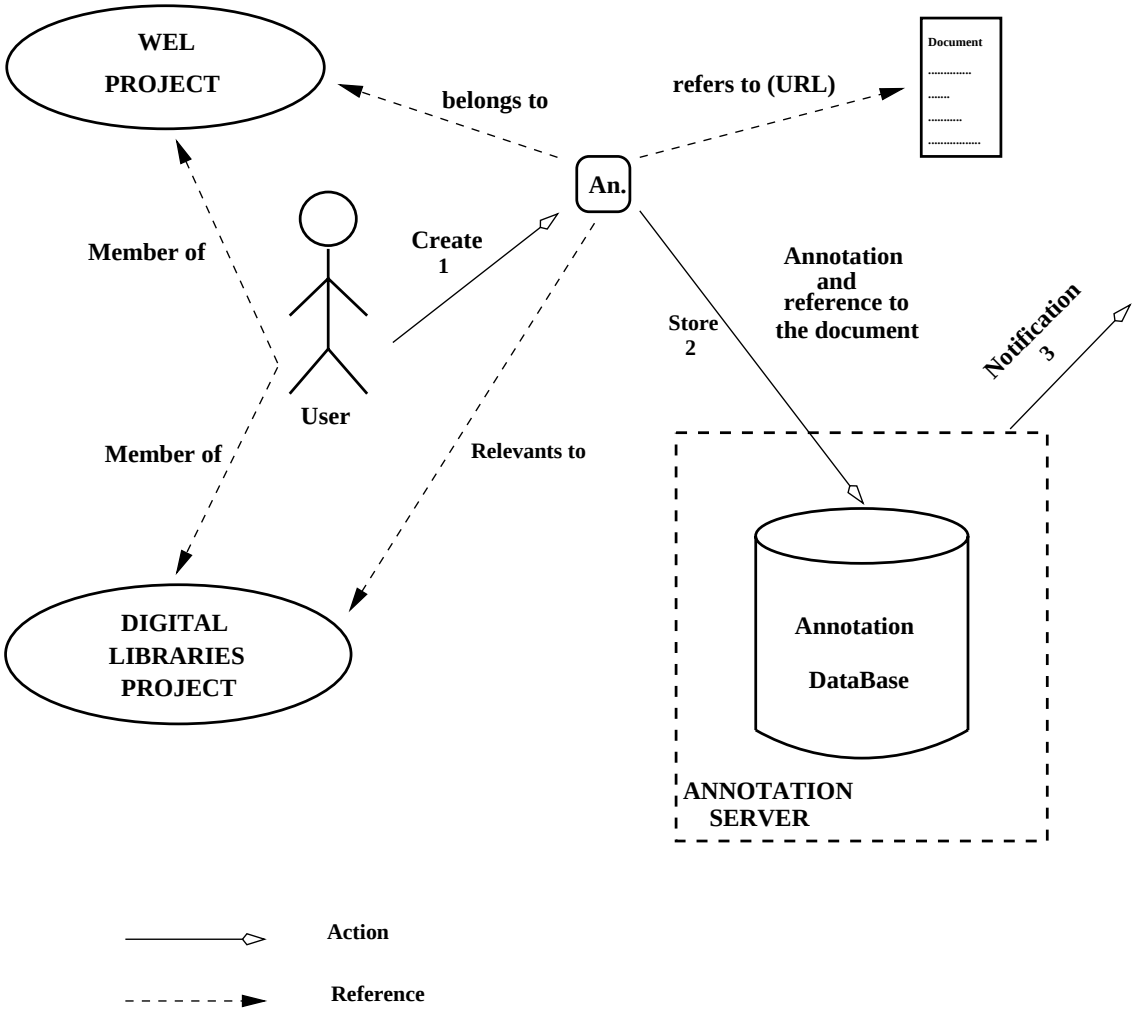

Figura 5.22: Esempio di creazione di un'annotazione.

Infine l'annotazione ed un riferimento al documento annotato (il suo URL) vengono memorizzati nel *Database*; è a questo punto che viene attivato il meccanismo di notificazione realizzato dall'*Annotation Server* e spiegato nella sezione 5.8.2.

5.8.2 Notificazione

Il meccanismo di notificazione è l'elemento che più caratterizza l'*Annotation Server* e lo rende un componente attivo all'interno del sistema, ovvero capace di compiere determinate azioni a seguito del verificarsi di eventi e nel rispetto di specifiche condizioni[11]. Per notificazione si intende

[11]Queste sono le cosiddette regole ECA-*Event Condition Action*.

la procedura attraverso cui gli utenti vengono portati a conoscenza di determinati cambiamenti[12] che sono avvenuti all'interno del sistema. La notificazione rappresenta una funzionalità di estrema importanza al fine della realizzazione di una struttura a supporto del lavoro cooperativo tra le persone appartenenti ad uno stesso gruppo di progetto od a gruppi di progetti differenti che si trovano nella stessa gerarchia.

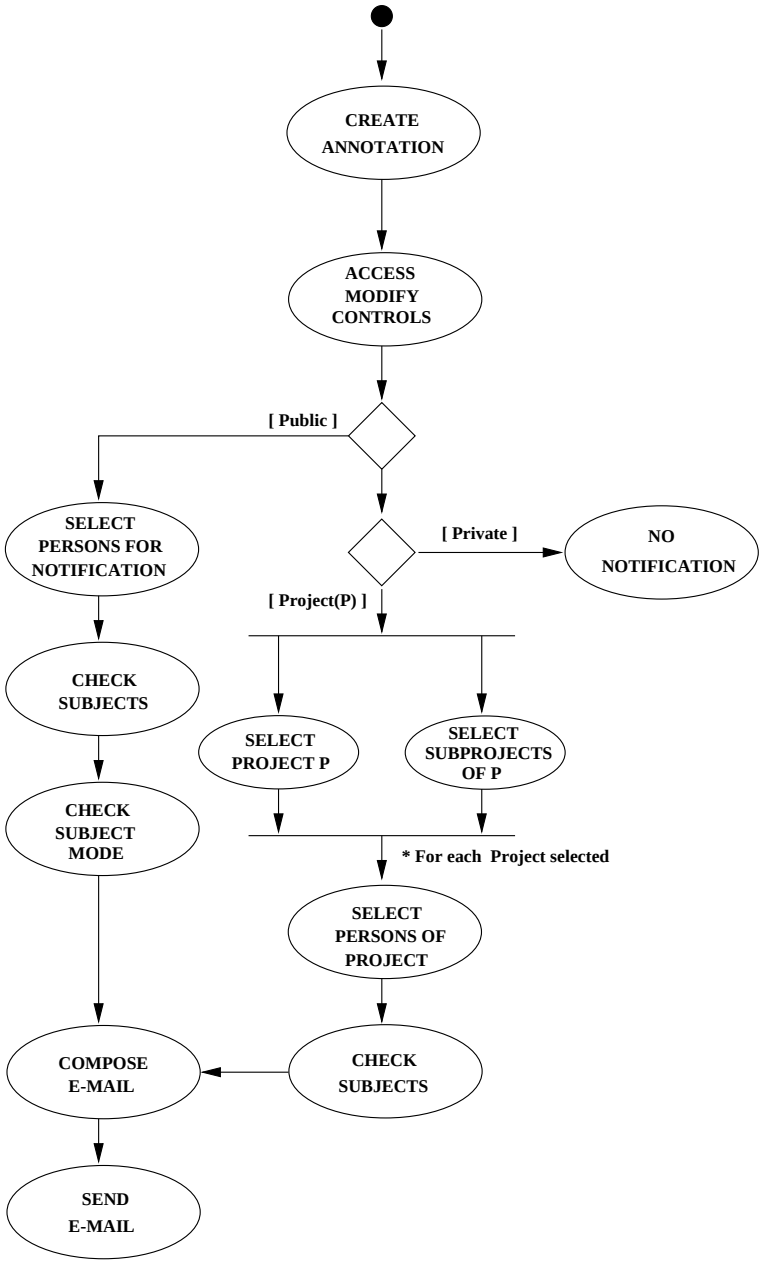

Figura 5.23: Esempio di *Activity Diagram*-Creazione/Notificazione.

[12] Con il termine cambiamenti si fa riferimento a tre precisi eventi: creazione di nuove annotazioni, cancellazione di annotazioni esistenti ed aggiunta o cancellazione di un utente da un gruppo di progetto.

In tale sezione viene preso in considerazione il meccanismo di notificazione a seguito di creazione di nuova annotazione; nella sezione 5.8.3 vengono esposte le regole di notificazione a seguito di cancellazione di una annotazione. La notificazione a seguito di aggiunta/cancellazione di un utente da un gruppo di progetto si basa su di una logica molto semplice e quindi non richiede assolutamente una sezione apposita: semplicemente tutti i membri del progetto ricevono un'*email* che li porta a conoscenza di quale componente è stato aggiunto/cancellato dal gruppo.

Nel momento in cui viene creata una nuova annotazione l'*Annotation Server* in primo luogo verifica l'*Access Modifier* specificato dall'utente che ha creato l'annotazione in questione. Se l'annotazione è *Private* non è necessaria alcuna notificazione in quanto nessun utente del sistema, all'infuori del suo autore, potrà accedere all'annotazione. Il meccanismo invece agisce nel momento in cui l'annotazione è *Project(P)* oppure *Public*.

Nel caso in cui l'annotazione è *Project(P)* riceveranno una notifica tutti gli utenti membri del progetto "P" (compresi i suoi sottoprogetti) interessati ad almeno uno dei temi a cui si riferisce l'annotazione. Ciò significa che la notifica è effettuata in dipendenza all'appartenenza dell'utente al progetto specificato ed al suo interesse ad un determinato tema oggetto di annotazione. Se vi sono membri del progetto "P" che non sono interessati a nessuno dei temi a cui si riferisce l'annotazione essi non riceveranno alcuna notifica ma saranno comunque abilitati a leggere l'annotazione. La figura 5.23 mostra l'*Activity Diagram* che rappresenta la funzionalità di notificazione a seguito di creazione di una nuova annotazione.

La notifica consiste in una *e-mail*, spedita dall'*Annotation Server* agli utenti opportunamente selezionati, contenente informazioni circa la nuova annotazione che è stata effettuata[13]. Se l'annotazione è *Public* l'algoritmo di notificazione funziona in maniera leggermente diversa; questo perché ogni utente registrato nel sistema appartiene, direttamente od indirettamente, al *Workspace Public* e quindi ci potrebbe essere il rischio che un utente riceva una quantità enorme di *e-mails* anche per annotazioni a cui non è interessato.

Per tale ragione nella figura 5.10 della sezione 5.3 è riportata la classe *Mode* attraverso la quale un utente può esplicitamente definire se un soggetto è, per lui, di interesse pubblico od è ad esso interessato solo nel contesto dei progetti di cui è membro. Ad esempio, facendo riferimento alla gerarchia riportata in figura 5.15, un utente appartenente al progetto *WEL* potrebbe essere interessato al soggetto *Indexes* in modalità *Public* ed al soggetto *ArtHistory* in modalità *No Public*. In tal modo l'*Annotation Server* invierà una notifica, a seguito di annotazione *Public* avente tra l'insieme dei suoi soggetti anche il soggetto *Indexes*, a tutti gli utenti del sistema interessati in base alla modalità *Public* ad almeno uno dei soggetti a cui si riferisce l'annotazione e quindi anche all'utente del progetto *WEL* sopra citato. Se invece l'annotazione *Public* comprende, nell'insieme dei suoi soggetti di riferimento, il soggetto *ArtHistory* l'utente in questione non riceverà alcuna notifica ma potrà comunque leggere l'annotazione effettuata.

La figura 5.24 schematizza il meccanismo di notificazione descritto. In tale esempio l'utente *User 1* effettua un'annotazione *Public* con *Subject B*. È questo un caso un po' anomalo, ma reso possibile dal sistema, in cui l'utente effettua un'annotazione con un soggetto diverso da quelli a cui è interessato.

L'*Annotation System* ed in particolare l'*Annotation Server* facendo uso dei dati presenti nel

[13]Tale particolare meccanismo di notificazione può essere definito notificazione *offline*.

Database verifica quali utenti devono essere notificati e quindi spedisce un'*email*. Nell'esempio gli utenti *User 2* e *User 4* riceveranno una notifica in quanto interessati in base alla modalità *Public* al *Subject B*. L'utente *User 3* invece essendo interessato al *Subject B* in base alla modalità *No public* non riceverà alcuna notifica.

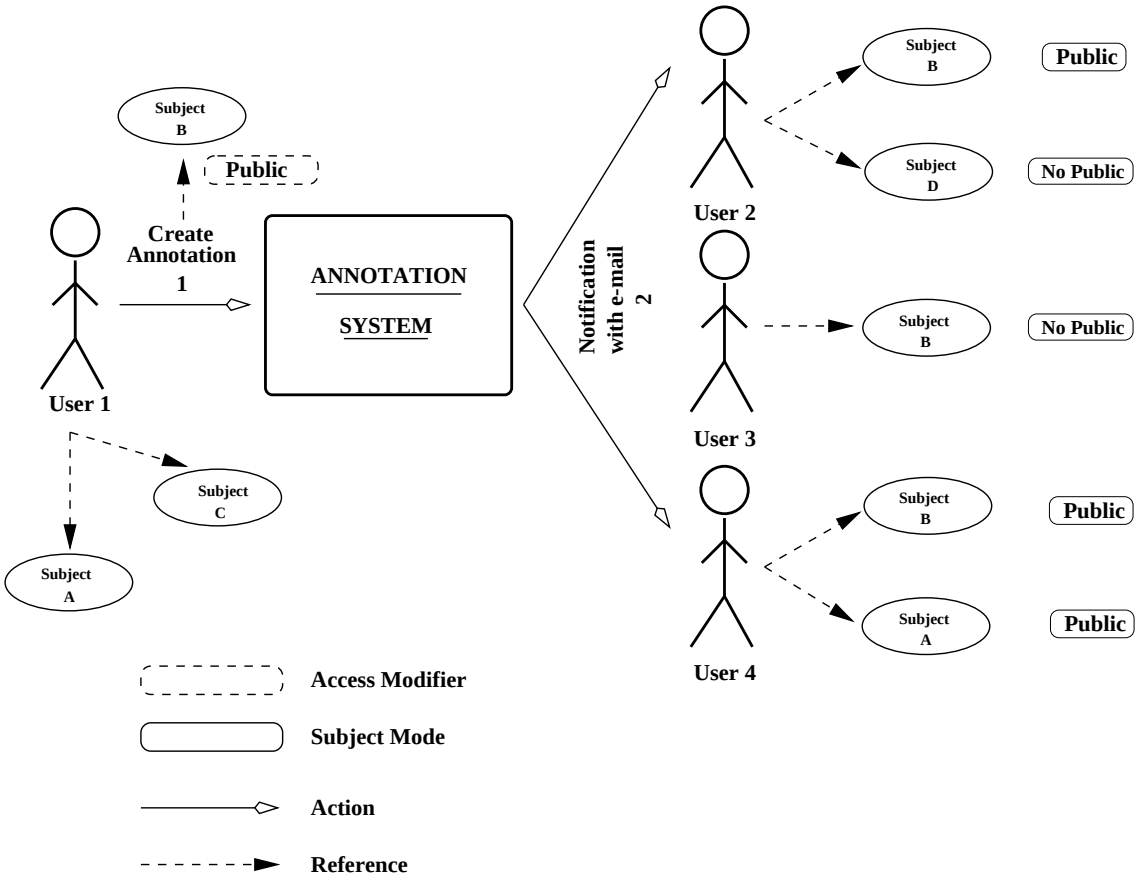

Figura 5.24: Meccanismo di notificazione.

5.8.3 Cancellazione

La cancellazione di annotazioni esistenti nel sistema è un meccanismo estremamente delicato e che pone rilevanti questioni circa le modalità in base alle quali deve essere effettuata la notificazione agli utenti interessati[14]. La figura 5.25 mostra l'*Activity Diagram* che rappresenta la serie di attività che vengono svolte dal sistema a seguito di cancellazione di un'annotazione.

Il problema principale riguarda la presenza di eventuali annotazioni articolate secondo una ben determinata gerarchia sottostante all'annotazione che si vuole cancellare. La decisione che è stata presa a tal riguardo prevede, in primo luogo, che solo il *System Administrator* può effettuare la cancellazione di determinate annotazioni. Il principio da seguire è che la cancellazione

[14]Gli utenti interessati sono tutti coloro che hanno effettuato un'annotazione che si riferisce direttamente od indirettamente all'annotazione che sta per essere cancellata.

è un'operazione da effettuarsi con una bassa frequenza in quanto, in ogni caso, porta alla perdita di contenuto informativo che può essere importante per gli utenti.

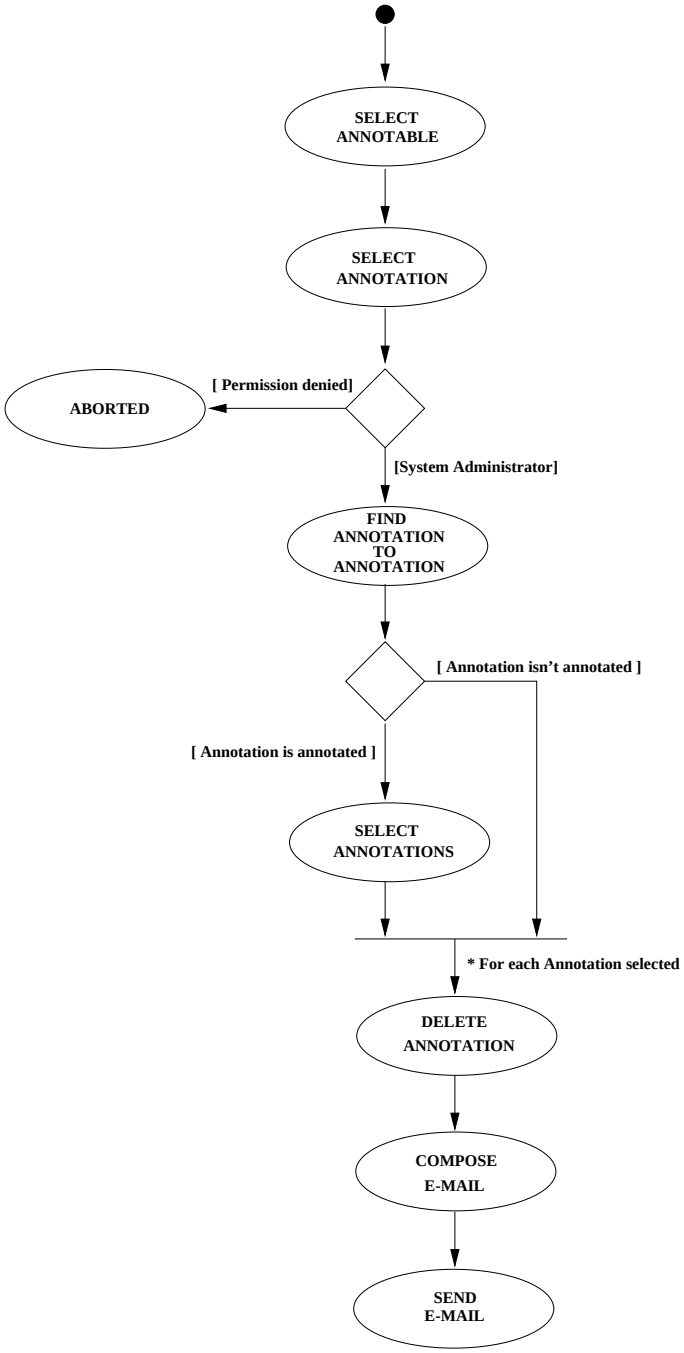

Figura 5.25: Esempio di *Activity Diagram*-Cancellazione/notificazione.

Esempi di situazioni in cui può essere richiesta la cancellazione di annotazioni sono l'eliminazione del documento originale oppure il fatto che l'annotazione risiede da lungo tempo nel

sistema ed il suo contenuto potrebbe non avere più significato. Nel caso in cui viene cancellato il documento il *System Administrator* , analizzato il contenuto delle annotazioni appartenenti alla gerarchia sottostante al documento, può decidere di effettuare delle cancellazioni; analogamente quando un'annotazione risiede per lungo tempo nel sistema essa può non aver più significato e quindi il *System Administrator* la può cancellare. Ciò che occorre sottolineare è che il *System Administrator* effettua tali cancellazioni in base alla **semantica** delle annotazioni ovvero analizzando il significato del loro contenuto.

Effettuata la cancellazione un primo problema che si pone è come si deve comportare il sistema con le annotazioni eventualmente presenti nella gerarchia sottostante all'annotazione eliminata. In tal caso si possono individuare due soluzioni principali:

- La gerarchia sottostante viene eliminata assieme all'annotazione inizialmente cancellata
- Le annotazioni che si riferiscono all'annotazione cancellata vengono riferite direttamente al documento originale sotto cui si struttura la gerarchia globale delle annotazioni; nel caso in cui il documento originale sia stato cancellato anche le annotazioni vengono cancellate.

La scelta dell'una o dell'altra soluzione dipende dalla semantica delle annotazioni. Ad esempio se vi sono annotazioni nella gerarchia sottostante che rappresentano un arricchimento del contenuto del documento originale la cancellazione significherebbe una perdita di contenuto informativo importante; d'altro canto se le annotazioni sottostanti rappresentano commenti all'annotazione cancellata riferirle direttamente al documento originale costituisce un errore concettuale abbastanza grave in quanto porterebbe ad inconsistenza della struttura informativa.

Nel modello proposto la soluzione adottata è la prima tra le due sopra descritte. Infatti la decisione è stata quella di cancellare automaticamente le annotazioni presenti nella gerarchia sottostante all'annotazione esplicitamente cancellata dal *System Administrator* integrando però tale automatismo con un meccanismo di notificazione effettuato sempre dall'*Annotation Server*.

Questa decisione è giustificata dal fatto che nella maggior parte dei casi le annotazioni che si strutturano in una gerarchia fanno riferimento ai contenuti presenti nelle annotazioni a cui si riferiscono; per questa ragione modificare la gerarchia porterebbe ad un'inconsistenza nelle informazioni; in tal caso è quindi preferibile cancellare tutte le annotazioni che si trovano al di sotto di quella originariamente eliminata. Presa però tale decisione si pone un nuovo problema consistente nello stabilire cosa deve essere notificato ai proprietari delle annotazioni cancellate ovvero quale deve essere il contenuto delle *e-mails* ad essi spedite. Anche tale questione può avere diverse soluzioni:

- La notifica può consistere in un semplice messaggio che mette al corrente della cancellazione e che riporta una copia dell'annotazione cancellata
- La notifica può consistere in una *e-mail* che riporta per ogni annotazione cancellata tutta la gerarchia soprastante sino all'annotazione originariamente cancellata dal *System Administrator*
- La notifica può consistere in un messaggio contenente una copia dell'annotazione cancellata e dell'annotazione che si trova nella posizione direttamente soprastante nella gerarchia di annotazioni

La soluzione implementata nel prototipo si ispira al terzo tra i principi sopra esposti. La ragione di tale scelta risiede nel fatto che nel momento in cui un'annotazione viene cancellata tutta la gerarchia che sta al di sotto di essa perde di significato e quindi viene eliminata. Al fine di salvare il contenuto informativo dell'intera gerarchia ogni autore delle annotazioni cancellate riceve una copia della propria annotazione e dell'annotazione che gli sta direttamente al di sopra. L'utente poi, ricevuta la notifica, deciderà autonomamente il da farsi eventualmente riferendo la propria annotazione ad altre annotazioni od al documento originale. L'autore dell'annotazione cancellata dal *System Administrator* riceverà invece un messaggio contenente la copia dell'annotazione cancellata e la copia, integrata con altrettanti messaggi di *warning*, di eventuali altre annotazioni di sua proprietà presenti nella gerarchia sottostante ed anch'esse eliminate automaticamente.

La figura 5.26 mostra schematicamente tale meccanismo.

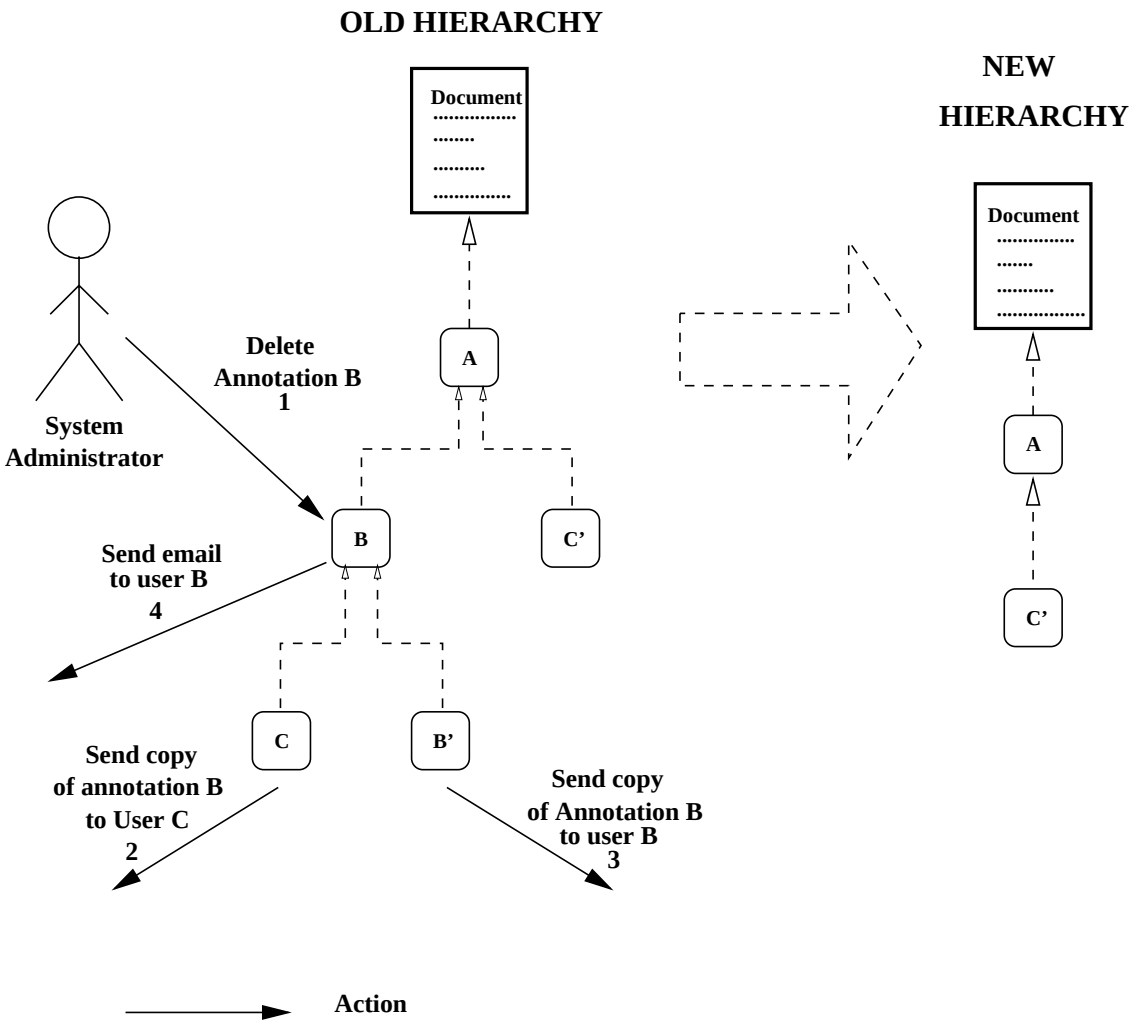

Figura 5.26: Esempio di cancellazione di un'annotazione.

Come spiegato in tale sezione la cancellazione delle annotazioni è un meccanismo estremamente complicato che scatena una serie di processi impossibili da gestire in modo univoco in

quanto dipendenti dalla semantica delle annotazioni. Per tale ragione si è deciso di dare la possibilità di cancellare le annotazioni solamente al *System Administrator* il quale dovrà gestire tale suo "potere" in maniera molto oculata ricorrendo alla cancellazione solo in casi necessari.

5.8.4 *Access Modifiers* delle annotazioni

Un ultimo aspetto di particolare interesse e che merita qualche considerazione è quello relativo alla possibilità offerta all'utente di effettuare annotazioni riferite ad altre annotazioni. Ciò determina la creazione di vere e proprie gerarchie di annotazioni che si sviluppano al di sotto di un determinato documento in base a logiche ben definite. Un esempio di gerarchie di annotazioni è riportato nella figura 5.27.

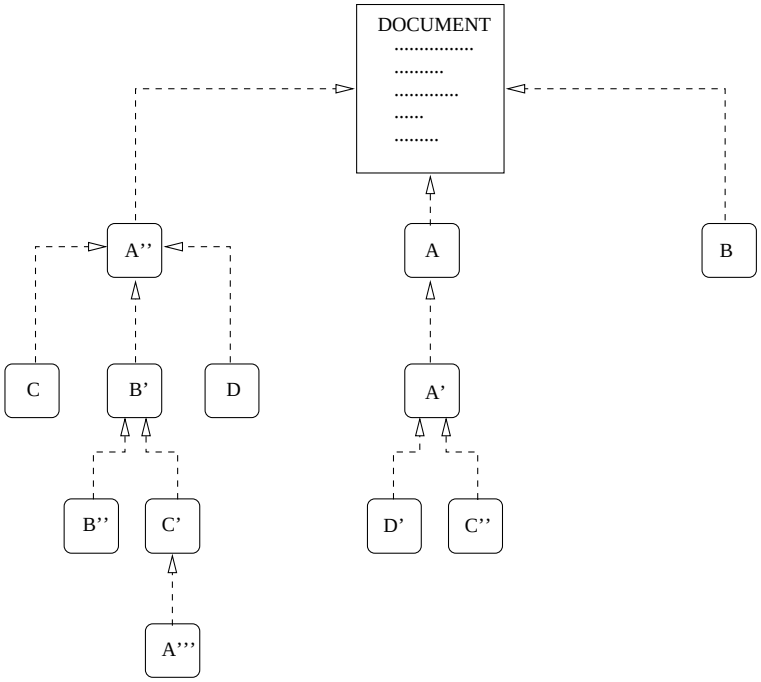

Figura 5.27: Esempio di una gerarchia di annotazioni.

In tale logica le annotazioni sono considerate come dei documenti HTML e come tali possono essere ulteriormente annotati[15]; occorre, a tal riguardo, fare alcune precisazioni relative agli *Access Modifiers* delle annotazioni.

Se un'annotazione è *Private* le annotazioni che stanno nella gerarchia sottostante saranno tutte *Private* dello stesso proprietario; questo è un concetto molto importante che si basa sul principio di riservatezza delle informazioni. Nel momento in cui un'utente crea una *Private Annotation* la sua volontà è quella di esprimere considerazioni personali che non vuole portare a conoscenza di nessun altro. Per tale ragione il sistema consente la lettura delle *Private Annotations* solamente al loro proprietario; la logica conseguenza è che la gerarchia di annotazioni

[15] In altre parole il sistema offre la possibilità di realizzare un meccanismo di annotazione ricorsiva.

che si struttura al di sotto di una *Private Annotation* sarà costituita interamente da annotazioni *Private* dello stesso proprietario dell'annotazione che costituisce il nodo radice.

Se un'annotazione è definita *Project (P)* si possono verificare diversi scenari. Le annotazioni nella gerarchia sottostante potranno essere *Project (P), Project(P')*, dove P' è un qualsiasi sotto-progetto di P, oppure potranno essere *Private* di uno qualsiasi degli utenti membri del progetto P.

Se un'annotazione è definita *Public* le annotazioni presenti nella gerarchia sottostante potranno essere anch'esse *Public, Project (P)* oppure *Private* di un qualsiasi utente registrato nel sistema. Nella figura 5.28 sono sinteticamente riportate le regole definite per il caso di annotazioni riferite ad altre annotazioni.

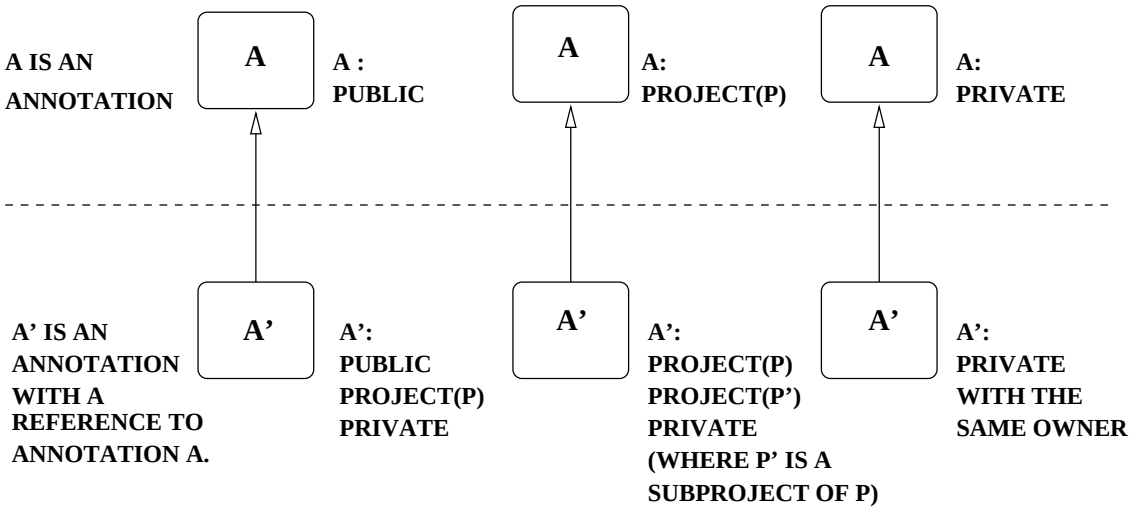

Figura 5.28: Annotatzioni ad annotazioni.

5.9 Componenti attive di un *Annotation Server*

Questa sezione propone una classificazione e descrizione di tutti quegli aspetti che presentano un comportamento attivo e che sono caratteristici del sistema modellizzato.

L'individuazione di tali compiti risulta essere estremamente importante in quanto suggerisce una possibile soluzione progettuale che prevede l'utilizzo di una base di dati attiva al fine della definizione dell'architettura del sistema e dell'implementazione del prototipo [Widom and Ceri 1996], [Ceri *et al.* 1997].

L'architettura proposta in tale testo (sezione 7.1) e la conseguente implementazione del prototipo (capitolo 7 e capitolo 8) possono essere realizzate facendo uso di una base di dati attiva. Da ciò ne consegue che tutte le funzionalità attive compiute a livello applicativo vengono integrate nel sistema di gestione della base di dati presente sull'*Annotation Server*.

In questo modo lo strato applicativo risulta essere più semplice in quanto costituito da un minor numero di funzioni; inoltre anche le prestazioni del sistema vengono migliorate grazie all'integrazione di molte attività al livello della base di dati. D'altro canto la realizzazione di

funzionalità applicative da parte del sistema di gestione della base di dati comporta una minor flessibilità dell'intero sistema.

Il seguito di tale sezione presenta una classificazione delle varie funzionalità identificate in fase di modellizzazione e che si possono configurare come regole attive.

Il contenuto di tale sezione deve essere integrato con la lettura della sezione 8.2 che riporta alcuni esempi in base ai quali è possibile implementare le regole attive di seguito esposte facendo uso dei *Trigger* [Corporation 1992a], [Corporation 1992b] messi a disposizione dal sistema di gestione della base di dati *Oracle v8.0* (sezione 6.2).

5.9.1 Principi generali

Il primo passo da compiere al fine di effettuare una classificazione esauriente dei vari comportamenti attivi presenti nel sistema di annotazione proposto consiste nel comprendere cosa si intende per regola attiva.

Le regole attive sono tutte quelle regole di produzione che seguono il cosiddetto paradigma Evento-Condizione-Azione [S.Ceri *et al.* 1996]. Una regola attiva si compone fondamentalmente di tre parti:

Evento per evento si intende tutto ciò che porta a dei cambiamenti nel sistema. Tipici eventi sono le modifiche effettuate alle informazioni contenute nella base di dati

Condizione è un predicato che può essere vero o falso ed il cui valore di verità viene valutato sulla base dei dati presenti nel sistema in un determinato momento

Azione è l'operazione, o l'insieme di operazioni che vengono eseguite dal sistema a seguito del verificarsi dell'evento e nel rispetto di determinate condizioni.

Il paradigma ECA (Evento-Condizione-Azione) riflette un comportamento semplice ed intuitivo: quando si verifica un evento, se viene soddisfatta una determinata condizione allora viene eseguita l'azione desiderata.

Un sistema in grado di supportare regole attive possiede un comportamento reattivo ed è realizzato in base al principio di indipendenza della conoscenza [S.Ceri *et al.* 1996], nel senso che la conoscenza racchiusa nelle regole attive non deve più essere posseduta dallo strato applicativo ma fa parte integrante della base di dati.

Il modello per il sistema a supporto dell'annotazione esposto in questo capitolo presenta una serie di comportamenti reattivi che possono essere raggruppati nel seguente modo:

- **Meccanismi di Notificazione:** sono tutti quei meccanismi esposti nella sezione 5.8.2 in base ai quali il sistema realizza una notificazione *offlines*, ovvero tramite la spedizione di *emails*, a tutti quegli utenti interessati al verificarsi di specifici eventi.

- **Compiti amministrativi:** queste attività riguardano sostanzialmente la gestione dei meccanismi di autentificazione degli utenti e l'amministrazione dei *login* effettuati presso il sitema.

- **Compiti gestionali:** per compiti gestionali si intende l'insieme di tutte quelle funzionalità che il sistema deve svolgere al fine di garantire una corretta gestione ed amministrazione dei progetti, degli utenti che vi fanno parte e delle annotazioni.
- **Gestione delle gerarchie di annotazioni:** un insieme di funzionalità che configurano un comportamento tipicamente reattivo del sistema è quello che comprende i compiti di gestione delle gerarchie di annotazioni che si possono generare con riferimento ad uno specifico documento WWW (sezione 5.8.4). Particolarmente significativo a tal proposito è il meccanismo di propagazione degli effetti che si instaura a seguito della cancellazione di una annotazione (sezione 5.8.3).

Ogni gruppo sopra definito è costituito da un insieme di funzionalità che si configurano come delle tipiche regole attive, ovvero che rispettano il paradigma Evento-Condizione-Azione e che possono essere quindi implementate come parte integrante di una base di dati attiva senza dover essere più gestite a livello applicativo (sezione 8.2).

Ultima considerazione a carattere generale, che deve essere fatta prima di procedere con la classificazione e descrizione dei vari gruppi di regole attive, consiste nel fatto che le annotazioni effettutate dagli utenti del sistema possono essere considerate tutte attive. Ciò significa che le operazioni effettuate sulle annotazioni, siano esse *Public Annotations, Project Annotations* oppure *Private Annotations*, costituiscono eventi che devono essere riconosciuti da un sistema attivo e che scatenano quindi una serie di reazioni nel rispetto di determinate condizioni.

Ovviamente gli eventi, le condizioni e le azioni che fanno parte di questo meccanismo attivo sono diverse a seconda della tipologia di annotazione.

Le sezioni che seguono propongono una classificazione formale delle varie regole attive identificabili nel modello proposto in funzione della tipologia delle annotazioni realizzate.

5.9.2 Meccanismi di notificazione

Nel modello astratto definito in tale capitolo la soluzione adottata è quella di realizzare i meccanismi di notificazione attraverso la modalità *off-lines*, ovvero per mezzo della spedizione di *e-mails* a tutti gli utenti interessati al verificarsi di determinati eventi interni al sistema (sezione 5.8.2).

I meccanismi di notificazione si configurano naturalmente come un gruppo di regole attive, ovvero come un insieme di funzioni che seguono il paradigma ECA (Evento-Condizione-Azione).

Gli eventi che attivano il meccanismo di notificazione sono i seguenti:

- Creazione di una nuova annotazione
- Cancellazione di un'annotazione
- Aggiunta di un utente ad un gruppo di progetto
- Eliminazione di un utente da un gruppo di progetto

Ognuno di tali eventi comporta la valutazione di determinate condizioni e l'esecuzioni di specifiche reazioni da parte del sistema.

Creazione di una nuova annotazione

Ogni utente registrato può creare delle annotazioni nell'ambito di un determinato progetto di cui è membro (sezione 5.8.1); al momento della creazione l'annotazione possiede automaticamente l'autore ed il progetto nell'ambito del quale è realizzata.

Di seguito l'utente deve specificare una serie di attributi della nuova annotazione, tra cui il testo, il fatto che sia riferita ad un documento od ad un'altra annotazione ma soprattutto gli *Access Modifiers* (sezione 5.4). Gli *Access Modifiers* sono le modalità[16] in base alle quali l'annotazione è realizzata e ne definiscono lo spazio di visibilità, ovvero l'insieme di progetti e quindi di utenti nell'ambito dei quali l'annotazione può essere letta e quindi eventualmente annotata con altre annotazioni.

La condizione che deve essere valutata nel momento in cui si verifica l'evento di creazione di una nuova annotazione consiste proprio nella determinazione della sua tipologia: se l'annotazione è *Private* non è necessaria alcuna notificazione, ovvero il sistema non scatena alcuna reazione all'evento che si è verificato nel sistema. Se invece l'annotazione è *Public* oppure *Project(P)* il sistema deve reagire con le seguenti azioni:

***Public Annotation*:** deve essere spedita un'*email*[17] a tutti gli utenti del sistema interessati in modalità *Public* (sezione 5.5.1) ad almeno uno dei soggetti (ambiti di ricerca) a cui fa riferimento la nuova annotazione

***Project(P) Annotation*:** deve essere spedita un'*email* a tutti i membri del progetto P, nonché dei suoi sottoprogetti, interessati ad almeno uno dei soggetti (ambiti di ricerca) a cui fa riferimento la nuova annotazione realizzata

Complessivamente l'insieme di regole attive che fanno parte di questo gruppo si possono schematizzare in questo modo:

1. **Evento:** Creazione di una nuova annotazione

2. **Condizione:** *Public Annotation or Project(P) Annotation*

3. **Azione:** Spedizione di *emails* ad un determinato gruppo di utenti

In riferimento a tale regola attiva occorre effettuare una considerazione importante; l'azione in effetti non consiste nella spedizione effettiva di un'*email* in quanto tale funzione può essere realizzata solamente a livello applicativo. Ciò che invece può essere fatto a livello di sistema di gestione di una base di dati attiva consiste nella determinazione del gruppo di indirizzi di posta elettronica di tutti quegli utenti a cui dovrà essere spedita un'*email*.

Tale considerazione vale ovviamente anche per gli altri esempi riportati di seguito.

[16] Le possibili modalità in base alle quali può essere scritta un'annotazione sono *Public*, *Project(P)* oppure *Private*.

[17] Questa *email* deve contenenere tutte le informazioni sufficienti al reperimento della nuova annotazione da parte degli utenti interessati.

Cancellazione di un'annotazione

La cancellazione di un'annotazione è un evento che deve essere trattato con particolare attenzione in quanto può scatenare nel sistema una serie di cancellazioni a catena. Questo avviene nel caso in cui sono presenti altre annotazioni che si riferiscono all'annotazione originariamente eliminata.

In questa sezione vengono considerati gli aspetti del meccanismo di notificazione in correlazione all'operazione di cancellazione di un'annotazione, rimandando alla sezione 5.9.5 la descrizione delle regole attive che gestiscono la propagazione degli effetti dell'operazione di cancellazione nell'eventuale gerarchia di annotazioni sottostanti.

Nel modello proposto la soluzione adottata è quella di dare solo al *System Administrator* la possibilità di cancellare annotazioni; il *System Administrator* ha la possibilità di cancellare qualsiasi annotazione presente nella base di dati senza alcun particolare vincolo o restrizione.

Ciò significa che a seguito dell'evento di cancellazione di un'annotazione non deve essere verificata alcuna condizione, o meglio che la condizione della regola attiva è configurabile come una condizione il cui valore di verità è sempre vero. Nel caso in cui si decida, in fase di modellizzazione, di gestire l'operazione di cancellazione in base a modalità differenti la regola attiva sotto riportata dovrà essere opportunamente modificata mediante la definizione di opportune condizioni la cui soddisfazione è il presupposto all'esecuzione dell'azione specificata.

L'azione che deve essere svolta dal sistema consiste nella spedizione di un'*email* al proprietario dell'annotazione contenente la copia dell'annotazione eliminata e l'informazione (URL) circa il documento a cui essa si riferiva. In sintesi la regola attiva sopra descritta può essere schematizzata nel seguente modo:

1. **Evento:** Cancellazione di un'annotazione
2. **Condizione:** Condizione sempre vera per definizione
3. **Azione:** Spedizione di un'*email* al proprietario dell'annotazione cancellata

Aggiunta di un utente ad un gruppo di progetto

Anche l'operazione per mezzo della quale si aggiunge un utente tra i membri del gruppo di un determinato progetto costituisce un evento che deve essere riconosciuto dal sistema al fine della realizzazione del meccanismo di notificazione.

A seguito dell'operazione di aggiunta di un utente ad un gruppo di progetto l'*Annotation Server* deve spedire un'*email* a tutti i componenti del progetto a cui è stato aggiunto il nuovo utente. La condizione della regola attiva che realizza tale procedura consiste quindi nella valutazione dell'appartenenza di un utente al progetto a cui è stato aggiunto un nuovo componente.

La regola attiva può essere schematizzata nel seguente modo:

1. **Evento:** Aggiunta di un utente ad un gruppo di progetto
2. **Condizione:** Appartenenza di un utente al gruppo di progetto a cui è stato aggiunto un nuovo membro

3. **Azione:** Spedizione di un'*email* all'utente selezionato in base al valore di verità della condizione specificata

Eliminazione di un utente da un gruppo di progetto

L'operazione di eliminazione di un utente da un gruppo di progetto è analoga a quella di aggiunta definita precedentemente; tutti i membri del progetto a cui è stato privato un componente vengono informati circa questa modifica attraverso la spedizione di un'*email*.

La regola attiva che realizza tale funzionalità può essere definita nel seguente modo:

1. **Evento:** Eliminazione di un componente da un gruppo di progetto
2. **Condizione:** Appartenenza di un utente al gruppo di progetto da cui è stato eliminato un componente
3. **Azione:** Spedizione di un'*email* all'utente selezionato in base al valore di verità della condizione specificata

5.9.3 Compiti amministrativi

I compiti amministrativi riguardano sostanzialmente tutte quelle funzionalità correlate alla gestione dei meccanismi di autentificazione degli utenti ed all'amministrazione dei *login* da essi effettuati. Queste funzioni si possono raggruppare nel seguente modo:

- Apertura di un *login*
- Cambiamento del contesto di lavoro
- Meccanismi di *Garbage Collection*

Anche le attività che riguardano i compiti amministrativi possono essere modellizzate sotto forma di regole che realizzano servizi attivi.

Apertura di un *login*

L'apertura di un *login* con il sistema consiste essenzialmente nella specifica da parte dell'utente del suo *userName*, della sua *password* e del progetto in cui intende lavorare. Nel caso in cui l'utente è membro di tale progetto e la sua *password* è corretta, il sistema deve determinare una chiave alfanumerica casuale che identifica univocamente il *login* e che viene memorizzata nella tabella *KEYS* (figura 5.20).

La regola attiva che realizza tale servizio può essere schematizzata in tal modo:

1. **Evento:** Creazione di un nuovo *login*
2. **Condizione:** Verifica della *password* e dell'appartenenza al progetto specificato
3. **Azione:** Determinazione e memorizzazione della chiave univoca da associare al *login*

Cambiamento del contesto di lavoro

Una volta effettuato il *login* nel contesto di un determinato progetto l'utente può decidere di cambiare il progetto in cui lavorare senza necessariamente fare un nuovo *login*. In tal caso se il nuovo progetto contiene tra i suoi membri l'utente in oggetto il sistema deve modificare la tupla che identifica il *login* con il riferimento al nuovo progetto, senza però modificare la chiave che resta comunque univoca e legata al *login*.

La regola attiva che realizza tale servizio si configura nel seguente modo:

1. **Evento:** Modifica del progetto di lavoro
2. **Condizione:** Verifica dell'appartenenza dell'utente al progetto
3. **Azione:** Modifica delle informazioni relative al *login* aggiornandole con i dati relativi al nuovo contesto di lavoro

Meccanismi di *Garbage Collection*

I meccanismi di *Garbage Collection* fanno riferimento sostanzialmente alle modalità in base alle quali è gestita la tabella *KEYS* presente nello schema riportato nella figura 5.20.

In tal caso la regola attiva è scatenata da eventi temporali, ad esempio giornalieri, a seguito dei quali viene controllato il campo *expires* di ciascuna tupla della tabella *KEYS*. Tale campo contiene la data in cui è stato effettuato il *login* rappresentato da quella specifica tupla.

Nel caso in cui il *login* è stato effettuato da un tempo superiore alla soglia prestabilita, la tupla viene cancellata dalla tabella. La regola attiva che realizza tale funzionalità si può schematizzare nel seguente modo:

1. **Evento:** Evento temporale (es. giornaliero)
2. **Condizione:** *Login* effettuato da un tempo superiore a quello stabilto come soglia
3. **Azione:** Cancellazione dalla tabella *KEYS* della tupla identificante il *login*

5.9.4 Compiti gestionali

Con il termine "compiti gestionali" si fa riferimento all'insieme di tutte quelle funzionalità che il sistema deve prevedere al fine della gestione delle strutture logiche identificate in fase di definizione del modello. Queste attività si configurano tipicamente come regole attive e possono essere realizzate direttamente facendo uso di una base di dati attiva, semplificando notevolmente l'insieme di compiti che devono essere svolti a livello applicativo.

Sostanzialmente si possono classificare tali funzionalità nei seguenti raggruppamenti principali:

- Gestione degli Utenti
- Gestione dei Progetti

- Gestione delle annotazioni

Di seguito vi è la descrizione delle funzioni che realizzano specifici compiti gestionali e che si configurano come delle tipiche regole attive.

Gestione degli Utenti

Le funzionalità di gestione degli utenti definite in fase di modellizzazione riguardano sostanzialmente gli aspetti relativi alla registrazione e cancellazione degli utenti.

Per quanto riguarda l'operazione di cancellazione degli utenti dal sistema, essa non si configura come una tipica regola attiva. Questo perchè tutto ciò che deve essere fatto a seguito di cancellazione di un utente è l'eliminazione dei suoi legami con i soggetti di interesse e con i progetti di cui fa parte o di cui ne è amministratore. Ma ciò può essere gestito con una normale base di dati relazionale, specificando i vincoli di integrità referenziale sulle tabelle e le modalità di gestione dei vincoli interrelazionali [S.Ceri *et al.* 1996].

La registrazione dell'utente è una procedura che si configura come una regola attiva scatenata dall'evento di aggiunta di un nuovo utente alla base di dati. In tal caso automaticamente l'utente deve essere aggiunto al *Workspace Public* ed ad esso deve essere assegnato il soggetto *Public* di tale progetto (sezione 5.5.3).

In sintesi la regola attiva si struttura nel seguente modo:

1. **Evento:** Registrazione di un nuovo utente
2. **Condizione:** Condizione sempre vera per definizione
3. **Azione:** Assegnazione al *Workspace Public* con soggetto *Public*

Altra operazione che segue il paradigma ECA (Evento-Condizione-Azione) è quella relativa all'aggiunta di un utente all'insieme dei componenti che costituiscono il gruppo di un determinato progetto. In tal caso oltre al meccanismo di notificazione descritto nella sezione 5.9.2 il sistema deve reagire assegnando all'utente i soggetti che fanno riferimento al progetto a cui è stato aggiunto. La regola attiva che realizza tale funzionalità può essere definita nel seguente modo:

1. **Evento:** Aggiunta di un utente tra i membri di un progetto
2. **Condizione:** Condizione sempre vera per definizione
3. **Azione:** Aggiunta dei temi del progetto al gruppo di soggetti di interesse del nuovo utente

Gestione dei Progetti

Per quanto riguarda le funzionalità di gestione dei progetti l'evento che merita maggior attenzione, e che scatena una serie di reazioni da parte del sistema, è quello relativo alla cancellazione di un determinato progetto dalla gerarchia esistente.

Anzitutto il progetto può essere eliminato solamente dal suo amministratore; inoltre la condizione che deve essere soddisfatta affinché si possa procedere con la cancellazione è relativa al

fatto che non deve essere aperto nessun *login*, da parte di nessun utente, sul progetto che si vuole cancellare. Infine un progetto, per poter essere eliminato, non deve avere sottoprogetti.

Nel caso in cui le condizioni siano soddisfatte il sistema deve assegnare gli utenti che facevano parte di quel progetto al gruppo del progetto che si trova nella posizione gerarchica immediatamente superiore.

Inoltre devono essere ridefiniti gli attributi di tutte quelle annotazioni in precedenza scritte nell'ambito di quel progetto od in esso definite rilevanti.

Dal punto di vista concettuale si possono schematizzare le seguenti regole attive che realizzano i compiti descritti:

1. **Evento:** Cancellazione di un Progetto
2. **Condizione:** Condizione sempre vera per definizione
3. **Azione:** Assegnazione degli utenti al superprogetto del progetto cancellato

La seconda regola attiva si può definire nel seguente modo:

1. **Evento:** Cancellazione di un progetto
2. **Condizione:** Controllo *Access Modifiers* dell'annotazione appartenente al progetto cancellato od in esso rilevante
3. **Azione:** Ridefinizione degli attributi dell'annotazione che esprimono la sua appartenenza o rilevanza nell'ambito di un progetto

La seconda regola attiva rappresenta concettualmente una serie di regole attive che devono essere realizzate in funzione degli *Access Modifiers* dell'annotazione e del fatto che essa appartenga al progetto cancellato o sia in esso rilevante.

Più in dettaglio, se l'annotazione è *Private* essa viene ridefinita appartenente al superprogetto; in caso contrario devono essere valutate diverse alternative.

Se l'annotazione appartiene al progetto eliminato ma è rilevante nell'ambito di uno dei suoi superprogetti, essa viene ridefinita appartenente al superprogetto in cui risulta essere rilevante. Nel caso in cui l'annotazione appartiene al progetto eliminato ed è in esso rilevante, essa viene definita appartenente e rilevante nell'ambito del superprogetto facente capo al progetto cancellato.

Gestione delle annotazioni

Le attività che riguardano la gestione delle annotazioni comprendono sostanzialmente la creazione di una nuova annotazione, la cancellazione di annotazioni esistenti e le varie funzioni di ricerca.

Le varie modalità di ricerca non si configurano come servizi attivi; la cancellazione di un'annotazione è un'operazione molto complessa ed ad essa è dedicata la sezione 5.9.5.

Per quanto riguarda la creazione di un'annotazione nella sezione 5.9.2 sono classificate regole attive che realizzano il meccanismo di notificazione a seguito di creazione di una nuova

annotazione. Inoltre nel momento in cui viene creata un'annotazione essa assume automaticamente i soggetti del progetto a cui appartiene. Si può quindi schematizzare una regola attiva che si occupa di svolgere tale funzionalità senza il bisogno di doverla realizzare a livello applicativo:

1. **Evento:** Creazione di un'annotazione
2. **Condizione:** Condizione sempre vera per definizione
3. **Azione:** Assegnazione dei soggetti di interesse del progetto alla nuova annotazione

5.9.5 Gestione delle gerarchie di annotazioni

Uno degli aspetti particolarmente complessi del modello proposto riguarda la gestione della gerarchia di annotazioni che si riferisce all'annotazione che si vuole eliminare. La sezione 5.8.3 riporta una descrizione dettagliata di questo problema e soprattutto le motivazioni che giustificano la scelta di cancellare, a seguito di eliminazione di un'annotazione, anche tutte le annotazioni che si trovano nella gerarchia sottostante.

Nella sezione 5.9.2 è descritta la regola attiva che realizza il meccanismo di notificazione a seguito di cancellazione di un'annotazione. Accanto a tale regola occorre definire un nuovo servizio attivo che si occupa di propagare una serie di cancellazioni a catena di tutte le annotazioni presenti nell'albero gerarchico. L'evento che scatena tale meccanismo è la cancellazione dell'annotazione; la condizione riguarda la verifica che vi siano annotazioni che si riferiscono all'annotazione eliminata; la reazione del sistema consiste nella cancellazione di tali annotazioni.

La schematizzazione concettuale di tale regola è la seguente:

1. **Evento:** Cancellazione di un'annotazione
2. **Condizione:** Annotazione riferita all'annotazione cancellata
3. **Azione:** Cancellazione dell'annotazione individuata

La combinazione di tale regola ricorsiva con quella definita nella sezione 5.9.2 determina l'eliminazione di un determinato ramo dell'albero gerarchico delle annotazioni e la spedizione di *emails* ai proprietari delle annotazioni cancellate.

Tale sezione ha avuto lo scopo di presentare una classificazione dei vari gruppi di regole che realizzano i servizi attivi identificati sulla base del modello realizzato. La sezione 8.2 riporta alcuni esempi di implementazione di tali regole utilizzando *Trigger Oracle* [Corporation 1992a], [Corporation 1992b].

Con tale argomento può considerarsi conclusa la trattazione del modello. Per una più completa comprensione degli argomenti trattati si consiglia la consultazione dei diagrammi riportati nell'appendice B, la lettura del capitolo 6 relativo alle tecnologie adottate nonché del capitolo 7 relativo all'architettura proposta ed al prototipo realizzato.

Capitolo 6

Tecnologie e metodi di sviluppo

Lo sviluppo di tecnologie sempre più evolute consente la realizzazione di sistemi in grado di perfezionare e migliorare i meccanismi e le modalità in base alle quali le persone svolgono il loro lavoro quotidiano.

Questo capitolo ha lo scopo di descrivere, in modo sintetico, le tecnologie che si sono utilizzate nello sviluppo del sistema a supporto dell'annotazione di documenti multimediali distribuiti sulla rete. Verranno presentati alcuni concetti fondamentali del linguaggio Java v1.1 sviluppato dalla Sun Microsystems con particolare riferimento ai meccanismi JDBC ed RMI (*Remote Method Invocation*) che sono stati utilizzati rispettivamente per la comunicazione con il *Database* e l'interfacciamento con il mondo esterno.

Verrà presentata inoltre la metodologia UML (*Unified Modeling Language*) utilizzata nella fase di modellizzazione, fase questa sempre molto delicata e che costituisce il punto cruciale nello sviluppo di un nuovo progetto.

6.1 Il linguaggio UML per la definizione di modelli astratti

Tale sezione ha lo scopo di presentare brevemente il linguaggio che è stato utilizzato in fase di modellizzazione del sistema. Naturalmente verranno posti in evidenza solamente gli aspetti semantici più importanti senza entrare in profondità nella descrizione degli aspetti sintattici del linguaggio UML.

Per avere una completa descrizione delle funzionalità offerte da tale linguaggio di modellizzazione, e poter comprendere quindi gli schemi riportati nel capitolo 5 relativo al modello proposto per il sistema di annotazione, il lettore può fare riferimento al testo *"UML distilled"* [Fowler 1997], oppure ai seguenti riferimenti [Booch and Rumbaugh 1995], [Booch and Rumbaugh 1997].

6.1.1 Introduzione

L'*Unified Modeling Language* (UML) è il seguito di tutta una serie di metodi per l'analisi e la progettazione orientata agli oggetti. L'UML è un linguaggio per la definizione di modelli e

non un metodo; i metodi consistono essenzialmente in un linguaggio di modellizzazione ed in un processo. Il linguaggio per la costruzione del modello è la notazione[1] che il metodo usa per esprimere il *design*. Il processo è invece l'insieme dei passi che devono essere compiuti per fare il progetto.

Il linguaggio UML attualmente definisce la notazione da utilizzare per il *design* ed il meta-modello da costruire. La notazione proposta è grafica e costituisce la sintassi del linguaggio.

Il primo passo da fare per giungere alla costruzione di un modello è la comprensione del dominio applicativo. Una delle tecniche disponibili per raggiungere efficacemente tale scopo è quella che prevede l'utilizzo di **use cases**[2]. Un *use case* è la descrizione di un aspetto di ciò che il sistema deve fare. La somma di tutti gli *use cases* fornisce una rappresentazione di come deve apparire il sistema esternamente. Una buona collezione di *use cases* è quindi centrale per capire ciò che in realtà l'utente desidera.

Nelle sezioni successive verranno descritti brevemente alcuni elementi del linguaggio UML iniziando dalla descrizione del processo che è stato seguito nello sviluppo del prototipo presentato.

6.1.2 Il processo

Come detto nella sezione precedente l'UML è un linguaggio per la costruzione di modelli e non prevede quindi la definizione di un processo che guida alla modellizzazione. Ciò perché l'UML può essere utilizzato con un qualsiasi processo essendo ad esso ortogonale.

La figura 6.1 mostra quale è stato il processo che si è seguito nello sviluppo del modello e nella sua implementazione, e nell'ambito del quale è stato utilizzato il linguaggio UML.

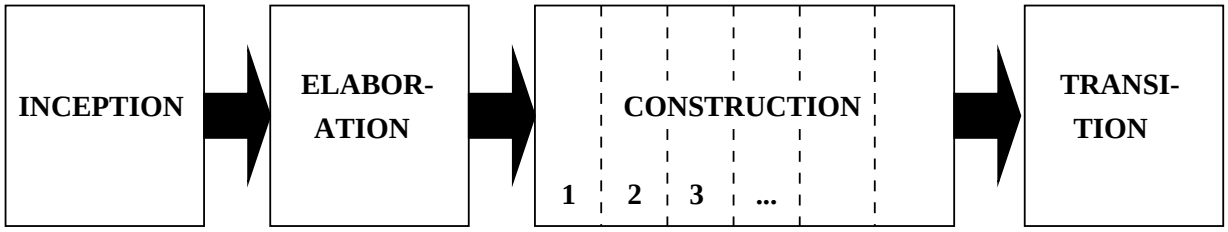

Figura 6.1: Processo di sviluppo.

Il processo di sviluppo del software adottato è un processo iterativo ed incrementale, in cui cioè il software non è realizzato in un sol colpo alla fine del progetto ma, invece, è sviluppato e realizzato a pezzi. Infatti la fase **Construction** è costituita da molte **Iterations**; in tale fase ogni iterazione produce del software, lo controlla e lo integra con il resto del sistema soddisfacendo un sottoassieme di tutti i requisiti richiesti al sistema. Le prime due fasi sono l'**Inception**

[1] Il più delle volte la notazione prevista dal linguaggio è grafica.

[2] Il dominio applicativo potrebbe essere descritto anche facendo ricorso a tecniche descrittive che utilizzano il linguaggio naturale ma che risultano meno intuitive e meno compatte rispetto a notazioni grafiche come quella degli *use cases*.

e l'**Elaboration**. Durante la fase di *inception* si definisce a grandi linee quale sarà il campo di applicabilità del progetto; in questa fase si ottiene l'impegno da parte del committente del progetto di realizzare il progetto stesso. Nella fase di *elaboration* si colleziona una serie di requisiti dettagliati, si effettua un'analisi di alto livello stabilendo l'architettura di base e creando il piano per la costruzione del prototipo. L'ultima fase è quella di **Transition** che include *beta testing, performance tuning* e *user training*.

Per avere una più dettagliata descrizione del processo brevemente descritto si può fare riferimento a [Booch 1994], [Gilb and Graham 1993].

6.1.3 *Use cases*

Nella sua essenza un *use case* è la descrizione di una tipica *interaction* tra l'utente ed il calcolatore. Ogni *use case* ha una serie di proprietà che si possono così riassumere:

- Un *use case* rappresenta l'identificazione di funzioni visibili dall'utente
- Un *use case* può essere semplice o complesso
- Un *use case* raggiunge un bisogno specifico dell'utente

Da quanto detto si capisce che per identificare gli *use cases* l'unico modo è quello di parlare con quelle persone che saranno gli utenti tipici del sistema cercando di capire le loro richieste.

Una distinzione importante a tal proposito deve essere effettuata tra ciò che si intende per **user goals** e ciò che invece si intende per **system interactions**. Per *system interactions* si intende l'insieme di tutte le modalità attraverso cui l'utente può comunicare con il sistema; gli *user goals* consistono invece in ciò che l'utente vuole fare con il sistema sviluppato. Il linguaggio UML prevede la possibilità di realizzare i cosiddetti **use case diagrams** molto utili per la costruzione degli *use cases*. In tali diagrammi si definisce **actor** il ruolo che l'utente ha rispetto al sistema. Sono gli *actors* le risorse principali di *use cases*; un singolo *actor* può eseguire più *use cases* e a sua volta un singolo *use case* può essere eseguito da più *actors*. Un *actor* non coincide necessariamente con una persona fisica ma può benissimo rappresentare anche un'altro sistema informatico che richiede lo svolgimento di funzionalità da parte del sistema che si sta correntemente modellizzando.

Gli *use cases* rappresentano quindi le funzionalità richieste esternamente. Ci può essere anche la possibilità che un *use case* non si riferisca ad un ben definito *actor*[3]. Una buona risorsa per identificare gli *use cases* sono gli eventi esterni. Inoltre nel linguaggio UML possono essere specificati *use cases* che **estendono** e che **utilizzano** altri *use cases*. Si ha *estensione* quando un *use case* descrive una variazione ad un normale comportamento; si ha *utilizzo* quando un *use case* fa parte di molti altri *use cases*.

Per una più estesa trattazione e spiegazione degli *use cases* riferirsi a [Fowler 1997], [Gilb and Graham 1993].

[3] In tale caso lo specifico *UseCase* rappresenta una generica funzionalità richiesta al sistema. Un esempio è l'applicazione che realizza automaticamente la cancellazione delle tuple obsolete della tabella KEYS (sezione 7.1).

6.1.4 *Class Diagrams*

La tecnica dei *class diagrams* è veramente fondamentale in una metodologia ad oggetti. Un **class diagram** descrive sostanzialmente il tipo di oggetti presenti nel sistema ed i vari tipi di relazioni statiche che esistono tra di essi. Si possono identificare due tipi principali di relazioni statiche:

Associations: rappresentano relazioni tra istanze di classi. Ogni *associations* può avere due ruoli, un ruolo per ogni direzione dell'*association*; i ruoli dell'associazione sono opzionali.

Subtypes: fanno riferimento sostanzialmente al concetto di gerarchie di generalizzazioni.

I *class diagrams* inoltre mostrano gli attributi e le operazioni offerte da una classe nonché le regole di visibilità che si vogliono stabilire per gli oggetti istanze di una determinata classe.

Come riportato in [Fowler 1997] ci sono tre prospettive in base alle quali un *class diagram* può essere costruito:

Conceptual: in base a tale prospettiva vengono rappresentati i concetti che caratterizzano il dominio applicativo;

Specification: tale logica guarda al software ma più precisamente all'interfaccia che deve essere offerta e non all'implementazione;

Implementation: in base a tale prospettiva si fa riferimento all'implementazione vera e propria delle varie classi.

Integrati con i *class diagrams* si possono poi utilizzare i *CRC-cards (Class-Responsibility-Collaboration)* [Beck 1989]. I *CRC-cards* consentono di definire con chiarezza ciò che ogni classe deve fare ed, eventualmente, con quali altre classi deve collaborare.

Tra le nozioni che stanno alla base dei *class diagram* ci sono poi quelle di *generalizzazione* e di *costraint rules* che indicano rispettivamente le gerarchie tra le varie classi ed i vincoli che devono ripettare gli oggetti che istanziano le classi. In UML i vincoli vengono rappresentati tra le parentesi graffe.

I *class diagrams* sono la spina dorsale di una metodologia ad oggetti e quindi devono essere usati sempre quando se ne sente la necessità in fase di costruzione del modello. La tecnica dei *class diagrams* può poi essere arricchita con una serie di notazioni addizionali. Tra queste l'idea di *stereotipo* introdotta da [Wirfs-Brock *et al.* 1990].

6.1.5 *Interaction Diagrams*

Gli *interaction diagrams* sono modelli che descrivono come gruppi di oggetti collaborano in una serie di comportamenti. Tipicamente un *interaction diagram* definisce il comportamento di un singolo *use case* evidenziando quali oggetti vengono coinvolti e quali messaggi vengono scambiati. Ci possono essere due tipi di *interaction diagrams* detti rispettivamente *sequence diagrams* e *collaboration diagrams*.

Una delle caratteristiche principali degli *interaction diagrams* è la loro semplicità;

Gli *interaction diagrams* sono quindi utili al fine di presentare il comportamento di diversi oggetti all'interno di un singolo *use case* ma non servono a definire con esattezza i comportamenti precisi di ogni oggetto.

6.1.6 *Package Diagrams*

Uno dei problemi più difficili da risolvere nello sviluppo di un grosso sistema è come poterlo spezzare in una serie di più piccoli e semplici sistemi. In UML l'idea è quella di raggruppare le classi assieme in un'unità di più alto livello; questo meccanismo di raggruppamento è chiamato *package*. Un *package diagram* quindi è una struttura che mostra i gruppi di classi e le dipendenze tra i gruppi. I *package diagrams* sono uno strumento molto utile soprattutto in progetti molto complessi ed articolati.

6.1.7 *State Diagrams*

Gli *state diagrams* sono delle tecniche molto utilizzate per descrivere il comportamento dei singoli oggetti. Essi descrivono tutti i possibili stati in cui si può trovare un determinato oggetto. Ad ogni stato possono essere associate delle attività ed ad ogni transazione tra stati può essere associata una regola ECA (Event-Condition-Action). Le azioni hanno carattere istantaneo e quindi non sono interrompibili mentre le attività possono essere interrotte nel tempo. Nel momento in cui uno stesso oggetto presenta un insieme di comportamenti indipendenti è possibile utilizzare dei diagrammi di stato concorrenti, che consentono cioè di definire contemporaneamente più stati in cui si può trovare un oggetto.

6.1.8 *Activity Diagrams*

Questi diagrammi sono particolarmente utili in connessione con i flussi di lavoro e descrivono il comportamento che hanno molti processi paralleli. Il cuore di tali diagrammi è rappresentato dal concetto di attività che rappresenta un compito che deve essere fatto da un essere umano o dal calcolatore. Ogni attività può essere seguita da un'altra attività in una semplice sequenza. Gli *activity diagrams* possono gestire anche dei processi paralleli ed in questo si differenziano dai più semplici *flow-chart*; tali diagrammi sono inoltre utili anche per rappresentare programmi concorrenti.

6.1.9 *Deployment Diagrams*

Un *deployment diagram* mostra le relazioni fisiche tra componenti hardware e software di un sistema. Tali diagrammi sono molto utili per descrivere come gli oggetti si muovono in un sistema distribuito. Le *connections* tra i nodi mostrano i cammini di comunicazione attraverso i quali il sistema interagisce. I *components* rappresentano invece moduli fisici e codice; praticamente questi corrispondono ai *packages* di un *package diagram*.

6.2 Oracle *v8.0* / SQL-*Structured Query Language*

Oracle System 8 costituisce lo sviluppo di *Oracle 7* che alla fine del 1996 ha rappresentato sicuramente il DBMS, per la gestione di *Database* relazionali, più diffuso sul mercato. *Oracle 8* costituisce l'evoluzione del sistema relazionale verso un modello *object-oriented* mantenendo e migliorando tutte le caratteristiche del sistema precedente. Esso è disponibile per una grande varietà di piattaforme hardware e di sistemi operativi supportando modelli *Client/Server*, modelli distribuiti e processi paralleli; esso fornisce molte soluzioni per l'interoperabilità con altri prodotti relazionali e con *legacy systems*, oltre che avanzati servizi per la replicazione ed il *warehousing*. Oracle 8 supporta il linguaggio SQL ed anche il linguaggio PL/SQL, che è una versione procedurale di SQL.

6.2.1 Il linguaggio SQL

SQL è il linguaggio principale per la gestione dei dati nei *Database* relazionali. Nel prototipo presentato in tale testo si è adottato un modello relazionale per lo sviluppo del *Database* utilizzando il linguaggio SQL per la creazione delle tabelle e per le interrogazioni che sono state necessarie nello sviluppo dello strato applicativo di gestione dell'*Annotation Server*. SQL è acronimo di *Structured Query Language* ed è un linguaggio di interrogazione per le basi di dati relazionali realizzato presso il laboratorio di ricerca IBM di S. Jose in California nella seconda metà degli anni settanta. SQL non è solo un linguaggio di interrogazione ma contiene anche le funzionalità di un *Data Definition Language* (DDL) ed anche di un *Data Manipulation Language* (DML) [S.Ceri *et al.* 1996]. La diffusione di SQL è dovuta essenzialmente all'intensa opera di standardizzazione che si è concentrata su questo linguaggio.

La prima definizione di uno standard per il linguaggio SQL è stata promulgata nel 1986 dall'ANSI (*American National Standard Institute*) che è l'organismo nazionale americano per gli standard. Una seconda versione è stata promulgata nel 1992 ed ad essa si fa riferimento con il termine SQL-92 od SQL-2. Una terza versione del linguaggio SQL è stata da poco definita dall'organismo di standardizzazione ISO.

Nel prototipo proposto il linguaggio SQL è utilizzato per la gestione della base di dati; l'utente non utilizza direttamente i comandi SQL ma vi accede in maniera indiretta attraverso il meccanismo JDBC (sezione 6.4).

6.2.2 Costrutti elementari del linguaggio SQL

Il linguaggio SQL contiene al suo interno le funzionalità di un *Data Definition Language* e di un *Data Manipulation Language*. Ciò significa che SQL prevede un insieme di primitive per la definizione dello schema concettuale di una base di dati relazionale ed un insieme di primitive per la modifica dell'istanza di una base di dati.

Lo schema di una base di dati può essere definito utilizzando il costrutto *create schema* mentre ciascuna tabella viene definita utilizzando il comando *create table*. Nella definizione delle tabelle è possibile specificare i valori di *default* degli attributi, i vincoli intrarelazionali ed

interrelazionali. In aggiunta a tutto ciò il linguaggio SQL offre la possibilità di creare degli indici di ricerca che migliorano notevolmente le *performances* dell'intera base di dati.

Aspetto essenziale del linguaggio SQL è la possibilità di effettuare interrogazioni alla base di dati; il costrutto principale a tal riguardo è l'istruzione *select* per mezzo della quale si possono selezionare le tuple di una determinata tabella in base a diversi criteri di ricerca. SQL esprime le interrogazioni in modo dichiarativo , ovvero si specifica l'obiettivo dell'interrogazione e non il modo in cui ottenerlo. Le interrogazioni possono essere effettuate utilizzando operatori aggregati che operano su gruppi di tuple opportunamente selezionate; inoltre è possibile effettuare interrogazioni nidificate in cui le tuple selezionate dall'interrogazione principale vengono determinate sulla base del risultato di un'altra interrogazione.

Infine il linguaggio SQL mette a disposizione i normali costrutti *insert, update* e *delete* per la manipolazione dei dati contenuti nel *Database*, nel rispetto dei vincoli di integrità che possono essere definiti tra tabelle di uno stesso schema relazionale.

Una descrizione dettagliata dei costrutti offerti dal linguaggio SQL si può ritrovare in [S.Ceri *et al.* 1996].

6.3 Il linguaggio *JAVA*

Il linguaggio di programmazione che si è adottato in fase di implementazione del prototipo è Java versione 1.1 che costituisce l'evoluzione della versione 1.0 con l'aggiunta di alcuni concetti quali quello di *Inner Class* ed un diverso modello per la gestione degli eventi.

Java è un linguaggio nato da poco tempo con l'obiettivo di creare dei programmi che possono essere eseguiti e distribuiti lungo la rete. Java sostanzialmente rappresenta un radicale cambiamento del Web arricchendo profondamente l'aspetto di interazione con l'utente che i sistemi software precedenti hanno sempre trascurato [Swadley 1996]. Il linguaggio è stato sviluppato dalla Sun Microsystems e la versione beta è stata presentata nell'estate del 1995. Java è un linguaggio *object oriented* che presenta molte somiglianze con il C++; la più grande differenza consiste nel fatto che Java non supporta il meccanismo dei puntatori e questo lo rende completamente *platform independent* e più sicuro. Tutte queste caratteristiche permettono di realizzare il concetto di Macchina Virtuale Java[4].

La tecnologia Java non è però limitata solamente al mondo del *server Web*; essa ad esempio può essere utilizzata anche per lo sviluppo di *embedded systems*.

Come sopra accennato Java consente una maggiore interazione nel mondo Web; questo viene realizzato attraverso lo sviluppo di contesti interattivi sotto forma di software che può essere scaricato dalla rete ed essere eseguito su di un qualsiasi *host* dotato di un *Java Interpretation Environment*. Java inoltre sviluppa l'idea di codice eseguibile *platform-independent* rendendo così possibili nuove forme di animazione, interazione, computazione ed applicazioni distribuite. Java sostanzialmente trasforma la concezione del Web in modo tale che gli utenti possano avere la sensazione di creare qualcosa con tale strumento piuttosto che solamente navigare alla ricerca di documenti.

[4]Per ulteriori informazioni relative al concetto di Macchina Virtuale Java fare riferimento al sito http://java.sun-.com.

Nel prototipo si è data particolare attenzione agli aspetti del linguaggio Java legati essenzialmente al meccanismo RMI (*Remote Method Invocation*) ed al protocollo di comunicazione JDBC che rappresentano rispettivamente i mezzi di comunicazione attraverso cui l'*Annotation Server* si interfaccia con il mondo esterno e con il *Database* sottostante. Per maggiori dettagli a tal riguardo fare riferimento alle sezioni riportate in tale capitolo con lo scopo di fornire una breve spiegazione di JDBC ed RMI.

6.4 JDBC

Se si volesse descrivere in poche parole JDBC lo si potrebbe definire come **un'API Java per l'esecuzione di istruzioni SQL**[5]. JDBC in realtà è un marchio registrato anche se molto spesso viene considerato come acronimo di *Java Database Connectivity*. Esso consiste in una serie di classi ed interfacce scritte nel linguaggio di programmazione Java. JDBC fornisce un'API standard per lo sviluppo di *tool/database* ed offre la possibilità di scrivere applicazioni per basi di dati usando semplicemente API Java.

Utilizzando le JDBC API, in altri termini, non è necessario scrivere un programma specifico per un determinato tipo di *Database*; è possibile scrivere un unico programma, usando le JDBC API, capace di inviare istruzioni SQL ad un qualsiasi *Database*. Ciò che serve ovviamente è una via di comunicazione tra le applicazioni Java ed i diversi tipi di *Databases*; JDBC è il meccanismo per fare ciò.

6.4.1 Modalità di funzionamento

JDBC è un'interfaccia di basso livello, il che significa che è usata per invocare comandi SQL direttamente. Esso esegue molto bene tale compito, ed è facile da usare con altre APIs che consentono la connessione con il *Database*, anche se è stato progettato per essere la base sopra la quale possono essere costruite altre interfacce e strumenti di più alto livello; un'interfaccia di alto livello è "*user friendly*", caratteristica questa che non possiede JDBC.

La JDBC API supporta due diversi modelli per l'accesso al *Database* definiti rispettivamente **two-tier model** (modello a due livelli) e **three-tier model** (modello a tre livelli).

Nel modello a due livelli un'applicazione Java parla direttamente con il *Database*. Questo richiede ovviamente un JDBC driver che può comunicare con il particolare DBMS (*Database Management System*) che si intende utilizzare. La figura 6.2 mostra l'architettura del modello a due livelli.

Le istruzioni SQL definite dall'utente sono consegnate al *Database* ed il risultato di queste istruzioni viene rispedito all'utente grazie al ruolo di intermiedario assunto dal driver. Il *Database* può essere caricato su una qualsiasi macchina connessa alla rete e con cui l'utente può quindi collegarsi. In pratica questo è un modello *Client/Server* in cui la macchina dell'utente è il *Client* e la macchina su cui risiede il *Database* è il *Server*. La rete di comunicazione può essere anche un'intranet oppure la rete internet. Questo è il modello che si è deciso di adottare nello sviluppo

[5] In altri termini attraverso il meccanismo JDBC è possibile accedere ad una qualunque base di dati attraverso applicazioni scritte in linguaggio Java.

del prototipo proposto in tale testo; per maggiori chiarimenti fare rifermento al capitolo 8 ed alla sezione 7.1.

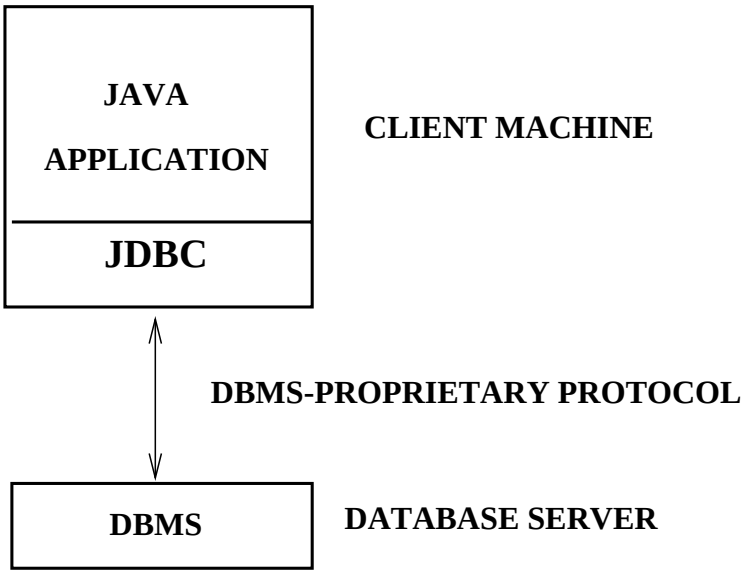

Figura 6.2: *JDBC/Two-tier model*.

Nel modello a tre livelli invece i comandi non sono spediti direttamente al *Database* bensì ad un "*middle tier*" il quale fornisce servizi e si occupa di spedire le istruzioni SQL al *Database*. Il *Database* quindi processa le istruzioni SQL ed invia i risultati allo strato di mezzo, il quale a sua volta invia i risultati all'utente. La figura 6.3 illustra l'architettura di questo secondo modello.

L'SQL (*Structured Query Language*) è il linguaggio standard per accedere ai *Database* relazionali. Un aspetto particolarmente intricato consiste nel fatto che sebbene molti DBMSs (*Database Management Systems*) utilizzino una forma standard per le funzionalità di base offerte dall'SQL, essi non sono compatibili con le recenti innovazioni riguardanti la sintassi e la semantica SQL. Una soluzione a tale problema è rappresentata dall'utilizzo delle JDBC API che permettono il trasferimento di *query* sotto forma di stringhe di caratteri che vengono inviate al driver DBMS sottostante. Ciò significa che un'applicazione è libera di utilizzare molte funzionalità dell'SQL correndo però il rischio di ricevere un errore dal DBMS che si occupa della gestione del DB. In tale contesto JDBC fornisce un utile supporto attraverso l'interfaccia *DatabaseMetaData* che consente di ottenere importanti informazioni relative al particolare DBMS utilizzato.

La tecnologia software necessaria per la realizzazione del meccanismo JDBC è parte integrante del pacchetto *JDK (Java Development Kit)* di dominio pubblico e disponibile sul sito *java.sun.com*. Javasoft fornisce tre componenti del prodotto JDBC come parti del pacchetto *Java Development Kit* (JDK):

- JDBC *driver manager*
- JDBC *driver test suite*

- JDBC-ODBC *bridge*

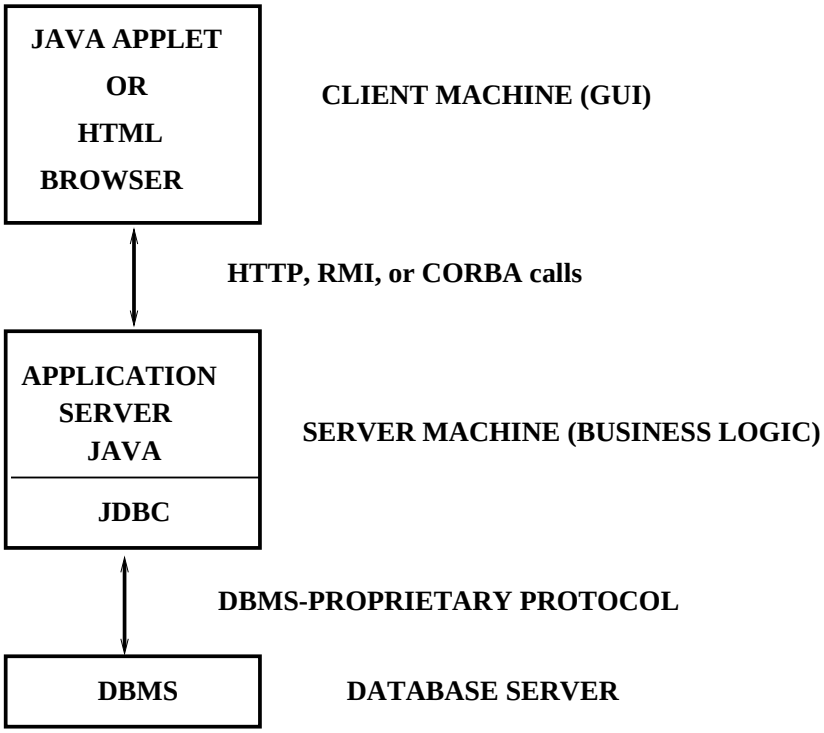

Figura 6.3: *JDBC/Three tier model*.

Il JDBC *driver manager* è la spina dorsale dell'architettura JDBC. Esso è molto semplice e la sua principale funzione è quella di connettere le applicazioni Java con il corretto JDBC driver permettendo così la comunicazione. Il JDBC *test suite* fornisce la certezza che il driver JDBC funzionerà con uno specifico programma; solo i programmi che superano positivamente il JDBC *Driver Test Suite* possono essere definiti JDBC COMPLIANT.

Il JDBC-ODBC *bridge* consente di utilizzare drivers ODBC come drivers JDBC. La figura 6.4 rappresenta schematicamente l'architettura brevemente descritta.

6.4.2 *Oracle JDBC Drivers*

Come spiegato nella sezione precedente JDBC è un insieme di classi e di interfacce scritte in Java che permettono ad altri programmi Java di inviare istruzioni SQL a DBMS di *Database* relazionali. Oracle fornisce due diverse categorie di JDBC drivers:

- **JDBC OCI**
- **JDBC Thin**

I drivers **JDBC OCI** sono di due tipi [6] e forniscono un'implementazione per le interfacce JDBC che utilizzano l'interfaccia OCI (*Oracle Call Interface*) per interagire con un *Database Oracle*. I drivers JDBC OCI sono stati realizzati per essere utilizzati da applicazioni scritte in linguaggio Java.

Nel prototipo è stato utilizzato il driver **JDBC Thin**[7] che impiega Java *sockets* per connettersi direttamente al *Database*. Tale driver fornisce una propria implementazione di una versione TCP/IP dell'*Oracle's SQL *Net*. Il driver è scritto completamente in codice Java ed è indipendente dalla piattaforma di sviluppo (*Platform Independent*).

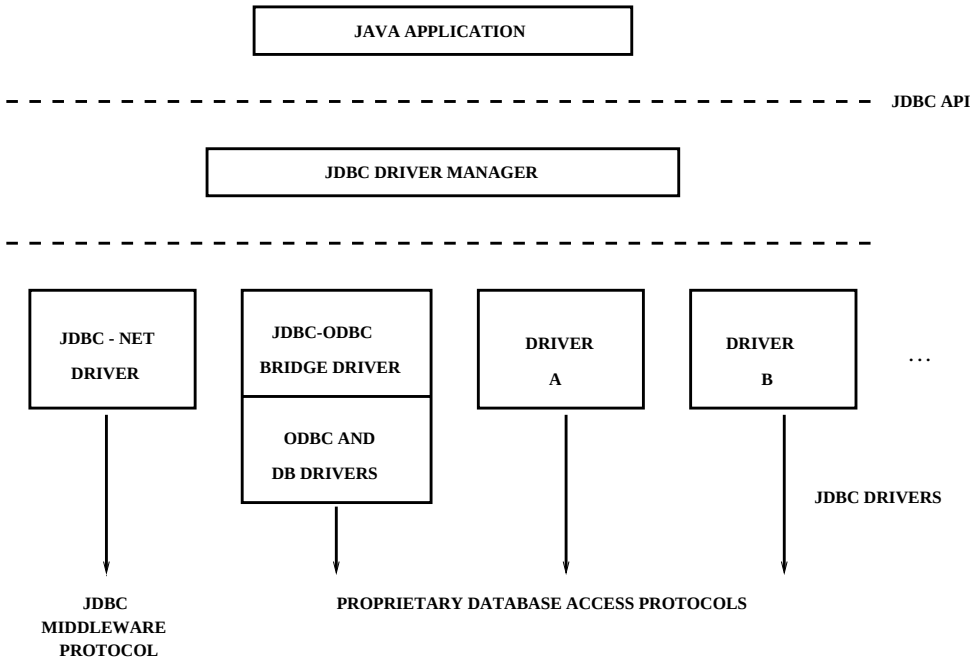

Figura 6.4: Architettura JDBC.

Per installare questo driver occorre creare una directory *jdbc* in quanto le classi in esso sviluppate accedono ai metodi utilizzando tale percorso. Occorre poi inizializzare correttamente le variabili d'ambiente al fine di poter identificare i *files* contenenti le classi e le librerie utilizzate dal driver. Compiute tali operazioni preliminari si possono identificare quattro passi che devono essere eseguiti da un programma scritto in Java affinché possa utilizzare un driver JDBC Oracle:

1. Caricare le classi JDBC includendo la linea di comando *import.java.sql.**

2. Caricare il driver includendo l'istruzione:

```
Class.forName(''oracle.jdbc.driver.OracleDriver'')
```

[6] Il diriver JDBC OCI7 deve essere utilizzato su di un *Client* che accede ad un *Database Oracle7* ed il driver JDBC OCI8 deve essere utilizzato per l'accesso ad un *Database Oracle8*.

[7] Il driver *JDBC Thin* viene utilizzato per accedere alla base di dati attraverso *Applets Java*.

3. Connettersi al *Database* includendo l'istruzione *DriverManager.getConnection(args)*

4. Sottoporre delle *query* utilizzando la sintassi JDBC standard ed ottenedo così i risultati desiderati

L'argomento che deve essere specificato nel metodo *getConnection* è l'URL (*Uniform Resource Locator*) del *Database* e deve essere scritto nella forma:
jdbc:oracle:drivertype:user/password@database
Per maggiori dettagli a tal riguardo fare riferimento al capitolo 8.

Estensioni offerte dal Driver *Oracle JDBC*

Quando si utilizza un *DBMS Oracle* il driver proprietario fornisce notevoli estensioni, ovvero notevoli funzionalità aggiuntive, rispetto al caso in cui si utilizzi il driver JDBC-ODBC *bridge*.

Tra queste estensioni vi è ad esempio la possibilità di ottenere direttamente un ROWID che può essere molto utile nella scrittura delle applicazioni; ancora la possibilità di definire dinamicamente il tipo delle colonne che costituiscono le tabelle e la possiblità di definire un *prefetching* delle righe di una determinata tabella. Attraverso il driver Oracle è inoltre possibile eseguire *query* a blocchi (*batching update*), possibilità questa che aumenta notevolmente le prestazioni dell'intero sistema in fase di accesso al *Database*.

Altra funzionalità fornita dal driver proprietario consiste nel fatto che è possibile creare degli *streams* ovvero dei flussi di trasferimento dei dati per i dati di tipo LONG o LONG RAW. In tale soluzione però è possibile ovviamente identificare delle restrizioni la prima delle quali è la perdita di portabilità del sistema, inevitabile nel momento in cui si adotta un driver proprietario anziché il driver standard ODBC. Inoltre utilizzando *drivers* proprietari si vincola il sistema ad utilizzare un determinato *Database* senza possibilità di effettuare cambiamenti in futuro.

Una descrizione di come tali funzionalità siano state utilizzate nello sviluppo del prototipo la si può avere consultando il capitolo 8 oppure analizzando il codice riportato nell'appendice C.

6.5 RMI-*Remote Method Invocation*

Il meccanismo RMI (*Remote Method Invocation*) offre al programmatore la possibilità di creare applicazioni Java distribuite, che richiamano cioè metodi presenti su altre macchine appese alla rete. In tale logica i metodi di oggetti remoti possono essere richiamati da un'altra macchina Java virtuale, possibilmente situata su di un'altro *host*. Un programma Java può effettuare una chiamata ad un oggetto remoto solo quando ha ottenuto un riferimento all'oggetto remoto stesso; questo riferimento può essere identificato in due modi:

1. utilizzando l'apposito metodo *Naming.lookup(AnnotationServer)* messo a disposizione dal servizio RMI;

2. ricevendo il riferimento come un argomento o come un valore di ritorno.

Con tale meccanismo un *Client* può chiamare un oggetto remoto situato su di un *Server*, ed il *Server* può anche essere un *Client* di un altro oggetto remoto. RMI utilizza il concetto di *Object Serialization* al fine di ordinare i parametri da trasmettere senza perdere informazioni riguardanti il loro tipo e supportando il concetto di polimorfismo tipico di un paradigma *object-oriented*.

6.5.1 Principi alla base del meccanismo RMI

È intuitivo il fatto che il meccanismo RMI può fallire per numerosi motivi legati essenzialmente ai problemi di comunicazione attraverso la rete ed ai problemi che si possono presentare sul *Server*. Per creare un oggetto remoto occorre prima definire un'interfaccia che verrà implementata da tale oggetto. Tale interfaccia deve possedere una serie di caratteristiche:

- deve essere *Public* in quanto un qualsiasi *Client* deve aver la possibilità di richiamare l'oggetto remoto che implementa l'interfaccia;
- deve estendere l'interfaccia *java.rmi.remote*;
- ogni metodo di tale interfaccia deve dichiarare le eccezioni *java.rmi.RemoteException* nell'apposita clausola.

I dettagli relativi all'implementazione dei metodi remoti, nel prototipo proposto, in base alla logica del meccanismo RMI sono riportati nel capitolo 8 e nell'appendice C.

I sistemi distribuiti richiedono che le computazioni eseguite in differenti spazi di indirizzamento delle variabili, potenzialmente su differenti *hosts*, siano in grado di comunicare tra di loro. Come meccanismo di base a supporto della comunicazione il linguaggio Java fornisce i cosiddetti *sockets*, i quali sono flessibili e rappresentano un ottimo strumento per comunicazioni a carattere generale. Lo svantaggio consiste nel fatto che i *sockets* richiedono che *Clients* e *Servers* siano coinvolti nella definizione di un protocollo di comunicazione a livello applicativo; ciò però è una procedura estremamente delicata e molto propensa ad errori. Un'alternativa ai *sockets* è il meccanismo RPC (*Remote Procedure Call*) il quale crea l'astrazione dell'interfaccia di comunicazione a livello di chiamata di procedura. In tal caso anziché lavorare direttamente con i *sockets* il programmatore ha l'illusione di chiamare delle procedure; il meccanismo RMI rispecchia completamente la logica di funzionamento della chiamata a procedure remote (*Remote Procedure Call*).

Il sistema *RMI-Java* segue fondamentalmente il paradigma *Client/Server*; in tale sistema distribuito un oggetto remoto possiede dei metodi che possono essere invocati da un'altra macchina virtuale, posta potenzialmente su di un altro *host*. *Remote Method Invocation* è essenzialmente l'azione attraverso cui si invoca un metodo di un'interfaccia remota implementata da un oggetto remoto. Ciò che è estremamente importante è il fatto che l'invocazione di un metodo remoto ha la stessa sintassi dell'invocazione di un metodo relativo ad un oggetto locale.

Il modello ad oggetti distribuiti differisce dal tradizionale modello ad oggetti nei seguenti punti fondamentali:

- i *Clients* degli oggetti remoti interagiscono con interfacce remote e non con le classi che implementano tali interfacce;

- gli argomenti non remoti sono passati per copia e non per indirizzo;
- un oggetto remoto è passato per indirizzo;

Le interfacce e le classi che vengono utilizzate dal meccanismo RMI sono riportate schematicamente nella figura 6.5 utilizzando la sintassi UML. Per una descrizione più dettagliata riguardante aspetti implementativi far riferimento al capitolo 8.

In tale modello distribuito i *Clients* interagiscono con gli *Stub Objects* i quali hanno esattamente lo stesso insieme di interfacce remote definite dalla classe che implementa l'oggetto remoto.

6.5.2 Architettura del sistema RMI

Il sistema RMI è costituito essenzialmente da tre strati:

- *stub-skeleton layer*
- *remote reference layer*
- *transport layer*

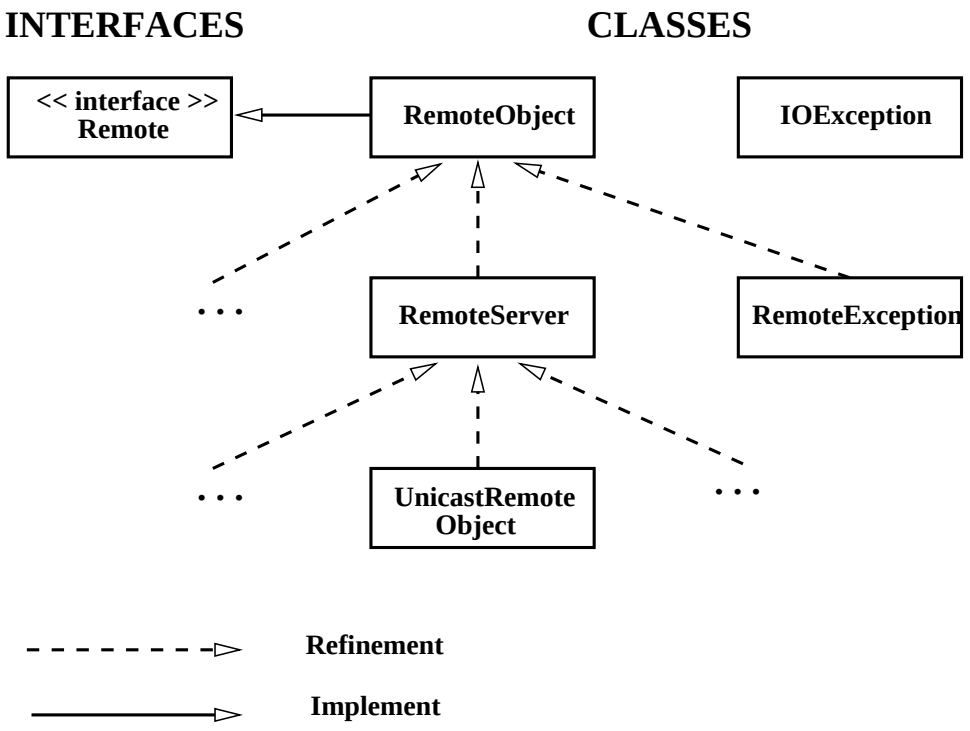

Figura 6.5: Classi ed interfacce del meccanismo RMI.

Il legame tra ogni strato è definito da una specifica interfaccia e da un determinato protocollo; per tale ragione ogni strato è indipendente dal successivo e può essere rimpiazzato da una qualsivoglia implementazione senza influenzare gli altri strati del sistema. La figura 6.6 mostra in generale l'architettura e le relazioni tra gli strati che la costituiscono.

Un'invocazione di metodo remoto proveniente da un *Client* e fatta ad un *Server* remoto attraversa dall'alto verso il basso gli strati dell'architettura RMI sul lato *Client* e quindi risale gli strati sul lato *Server*[8]. Un *Client* per poter invocare un metodo remoto deve ottenere un riferimento all'oggetto remoto che non è altro che un riferimento alla procedura *Stub* residente sul lato *Client*. Questa *Stub* è l'implementazione di un'interfaccia dell'oggetto remoto ed effettua una chiamata che viene inoltrata attraverso gli strati dell'architettura di figura 6.6.

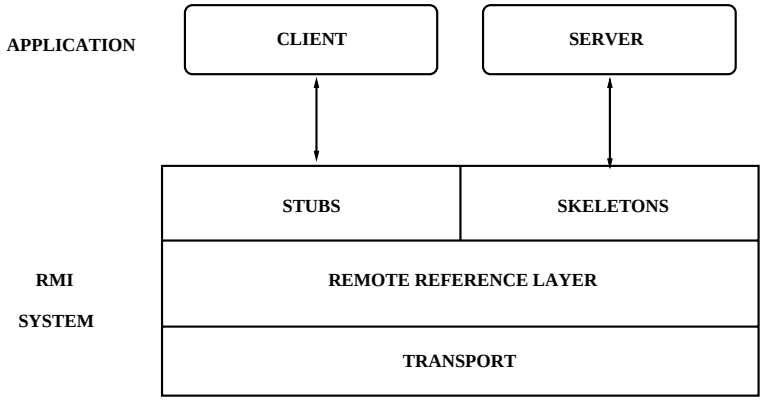

Figura 6.6: Architettura del sistema RMI.

Il *Remote Reference Layer* ha la responsabilità principale di trasportare fuori la semantica dell'invocazione. Il *Transport Layer* è invece responsabile di tutti quegli aspetti relativi alla gestione della connessione tra macchine remote. Lo *Skeleton* che risiede sul lato *Server* si occupa di effettuare delle chiamate alle implementazioni degli oggetti remoti invocando il metodo in questione, ovvero il metodo specificato dal *Client*. Questa è in breve la descrizione della tecnologia utilizzata per consentire la comunicazione tra l'*Annotation Server* ed un qualsiasi *Client* che desidera utilizzare il sistema di annotazione proposto in questo testo. Gli aspetti puramente implementativi relativi al meccanismo RMI nell'ambito del prototipo sviluppato sono riportati nel capitolo 8 e nell'appendice C.

[8]La logica di funzionamento è quella di un normale protocollo di comunicazione in rete come ad esempio il protocollo ISO OSI.

Capitolo 7

Architettura e funzionalità del prototipo

Questo capitolo si struttura fondamentalmente in due parti principali ciascuna delle quali presenta implicazioni e riferimenti incrociati che ne giustificano l'accorpamento a livello espositivo.

In primo luogo verrà descritta l'architettura del sistema sviluppato; tale descrizione sarà particolarmente dettagliata e l'attenzione sarà rivolta soprattutto all'architettura dell'*Annotation Server* il cui prototipo è presentato in tale testo.

Saranno inoltre esposti brevemente i principi di realizzazione e di funzionamento del *Proxy Server* e dell'interfaccia utente realizzata sul *Browser* attraverso cui l'utente può interagire con il sistema per l'annotazione proposto.

Il secondo passo consiste nella descrizione dei principi generali che hanno guidato l'implementazione dell'*Annotation Server*, dei meccanismi di comunicazione remota tra il *Browser Client* e l'*Annotation Server* e delle modalità in base alle quali l'*Annotation Server* accede alla base di dati. In tale contesto vengono definite le specifiche funzionali del prototipo.

La fase di implementazione del prototipo è la logica conseguenza di ciò che è stato definito e progettato attraverso la definizione del modello astratto (capitolo 5).

Il contenuto di questo capitolo è integrato dai concetti esposti nel capitolo 8, in cui è riportata la descrizione dettagliata di alcuni aspetti implementativi ritenuti particolarmente significativi; per meglio chiarire l'esposizione tale descrizione è integrata da alcune porzioni di codice che fanno riferimento ai temi trattati e che fanno parte dello strato applicativo residente sull'*Annotation Server* (sezione 7.1.3).

7.1 Architettura del sistema

In tale sezione viene presentata l'architettura del sistema sviluppato e riportata nella figura 7.1. I componenti principali che possono essere identificati sono i seguenti: *Annotation Server, Browser, Proxy Server* ed i meccanismi di comunicazione.

Nell'ambito dell'*Annotation Server* dal punto di vista architetturale si possono definire altri due componenti ovvero il *Database* e l'*Application Layer*.

Le comunicazioni tra i componenti principali dell'architettura di figura 7.1 avvengono attraverso il meccanismo *RMI-Remote Method Invocation* utilizzando il protocollo TCP/IP per lo

scambio di informazioni attraverso la rete. Il sistema interagisce con il *World Wide Web* ottenendo riferimenti a quei documenti che sono stati annotati.

Infine è anche contemplata la possibilità di realizzare un *FireWall* avente lo scopo di garantire la sicurezza dell'intero sistema proteggendolo da eventuali accessi esterni che, intenzionalmente, potrebbero essere diretti alla destabilizzazione dell'intera struttura.

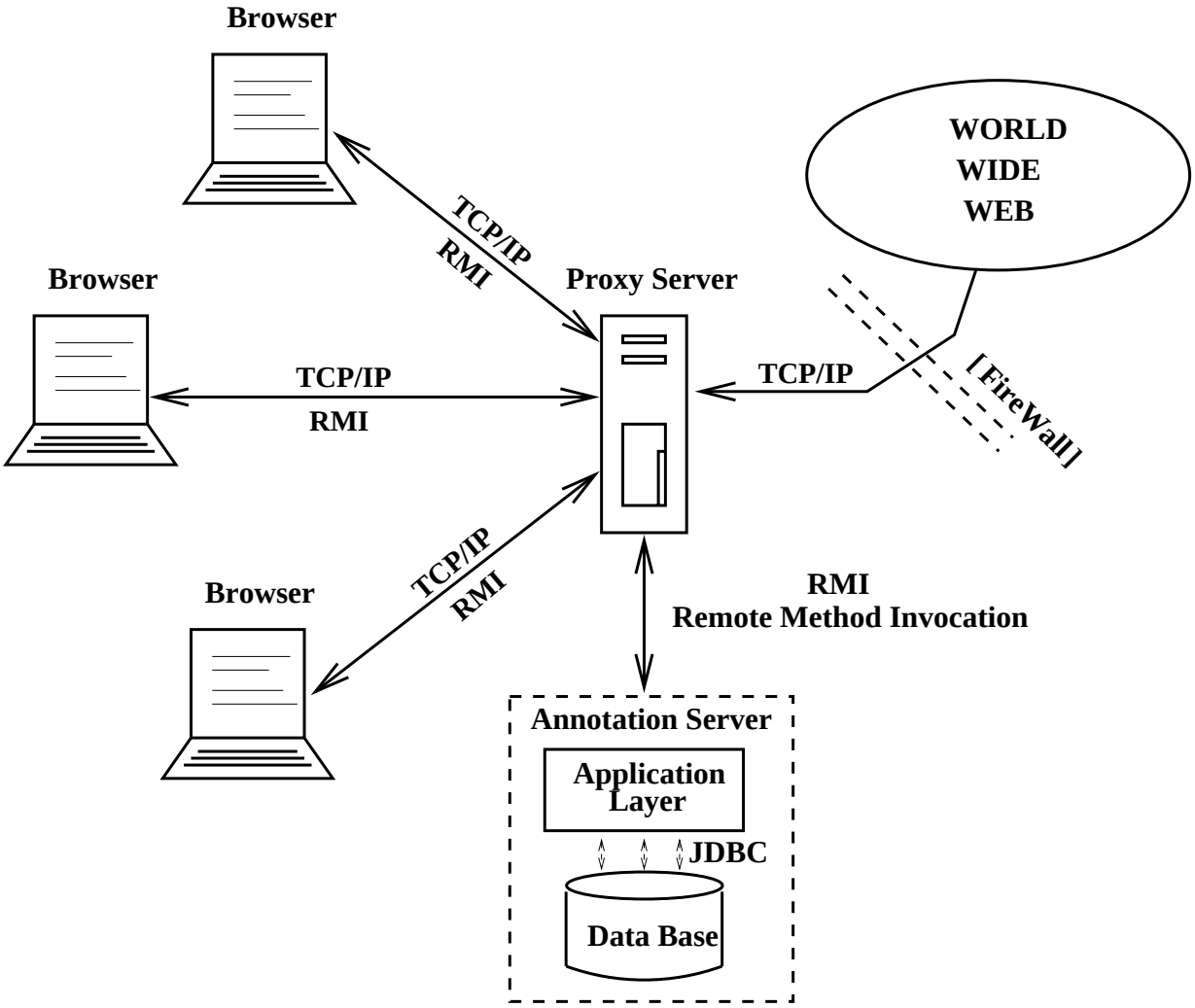

Figura 7.1: Architettura del sistema.

Di seguito è riportata una breve descrizione di ogni componente con particolare enfasi per l'*Annotation Server* e per i meccanismi di comunicazione le cui implementazioni e realizzazioni prototipali sono oggetto di tale testo.

L'implementazione del *Proxy Server* e dell'interfaccia grafica realizzata sul *Browser* costituisce oggetto di un altro lavoro di tesi in corso di sviluppo presso la Technische Universität Hamburg-Harburg nell'ambito del progetto *WEL Warburg Electronic Library* (sezione 4.2.2).

7.1.1 L'*Annotation Server*

L'*Annotation Server* è l'elemento centrale dell'intera architettura; come si può osservare dalla figura 7.1 l'*Annotation Server* amministra la base di dati utilizzando il meccanismo JDBC ed inoltre comunica con il *Proxy Server*; esso può quindi essere considerato un *Gateway* tra il *Proxy Server* e la base di dati. Lo strato applicativo presente sull'*Annotation Server*, descritto a livello generale nella sezione 7.1.3 e più in dettaglio nel capitolo 8, è costituito da un'ampia serie di metodi remoti sempre a disposizione del *Proxy Server* e che mantengono aggiornata la base di dati.

Sull'*Annotation Server* viene realizzato il meccanismo di autentificazione degli utenti ed i meccanismi di gestione degli errori. Tale componente inoltre reagisce attivamente alla creazione di nuove annotazioni, od alla cancellazione di quelle già esistenti, realizzando le procedure di notificazione discusse nelle sezioni 5.8.2 e 5.8.3.

L'*Annotation Server* è un componente attivo nell'ambito dell'intera architettura e potrebbe essere realizzato utilizzando una base di dati attiva attraverso cui implementare le regole attive che descrivono le varie funzionalità offerte da tale componente. Le sezioni 5.9 e 8.2 descrivono le modalità in base alle quali potrebbe essere possibile realizzare un'*Annotation Server* tramite l'organizzazione e l'implementazione di regole attive.

L'*Annotation Server* svolge una serie di compiti di fondamentale importanza e che possono essere classificati nel seguente modo:

Gestione degli Utenti l'aspetto relativo alla gestione degli utenti è certamente di primaria importanza e concerne essenzialmente l'organizzazione logica degli utenti in gruppi di progetto come descritto nel capitolo 5. L'*Annotation Server* inoltre si occupa di gestire il *Login* ed il *Logout* degli utenti che accedono al sistema tramite interfacce grafiche realizzate sui *Clients*.

Gestione dei Progetti sull'*Annotation Server* sono realizzate tutte le procedure necessarie per la gestione dell'albero gerarchico dei progetti. Tali compiti sono estremamente importanti in quanto hanno delle implicazioni rilevanti sulle modalità in base alle quali possono essere effettuate ed organizzate le annotazioni. Le logiche in base alle quali la struttura dell'albero dei progetti influenza le modalità di annotazione sono esposte nel capitolo 5.

Gestione delle Annotazioni l'*Annotation Server* fornisce un ampio insieme di metodi per la gestione delle annotazioni in termini di creazione, ricerca e cancellazione. In particolare sono offerti metodi di ricerca flessibili che offrono la possibilità all'utente di trovare rapidamente l'insieme di informazioni desiderate.

Meccanismi di Notificazione i meccanismi di notificazione sono quelli che più caratterizzano l'*Annotation Server* e lo rendono un componente attivo all'interno dell'architettura. Una descrizione dettagliata di tali meccanismi è riportata nella sezione 5.8.2.

Meccanismi di Autentificazione l'*Annotation Server* si occupa anche di realizzare la funzionalità di autentificazione degli utenti descritta nella sezione 7.2.

Gestione della base di dati la gestione della base di dati contenente le informazioni relative alle annotazioni, agli utenti ed ai progetti è un naturale compito che deve essere svolto dall'*Annotation Server*.

I metodi dell'*Annotation Server* sono stati realizzati al fine di garantire consistenza tra le informazioni contenute nella base di dati e gli oggetti che costituiscono la loro immagine a livello applicativo; tale scelta garantisce consistenza ma comporta una riduzione delle prestazioni a causa di continui accessi alla base di dati.

Gestione degli errori l'*Annotation Server* si occupa inoltre di gestire completamente gli errori logici che possono essere compiuti dall'utente del sistema. I messaggi di errore vengono propagati dallo strato applicativo all'interfaccia grafica realizzata sul *Browser* al fine di portarli a conoscenza dell'utente.

7.1.2 Il *Database*

La base di dati costituisce un componente interno all'*Annotation Server*; in essa vengono memorizzate tutte le informazioni relative agli utenti del sistema, ai progetti realizzati ed organizzati gerarchicamente ed alle annotazioni effettuate dagli utenti nell'ambito di determinati progetti.

La base di dati utilizzata è relazionale e lo schema concettuale progettato è riportato nella figura 5.19; i principi generali in base ai quali è stata implementata la base di dati sono riportati nella sezione 8.1.1. In fase di implementazione della base di dati si è prestata particolare attenzione all'esplicitazione di tutti i vincoli di integrità referenziale al fine di sfruttare i meccanismi messi a disposizione dal DBMS Oracle per la gestione degli errori.

7.1.3 L'*Application Layer*

Lo strato applicativo realizzato sull'*Annotation Server* prevede una serie di metodi remoti realizzati in base alle specifiche del meccanismo RMI secondo il paradigma di chiamata a procedure remote. Lo strato applicativo accede direttamente alla base di dati utilizzando il meccanismo JDBC (sezione 6.4).

Una descrizione dettagliata dei metodi che costituiscono lo strato applicativo dell'*Annotation Server* è riportata nella sezione 7.3.

7.1.4 Il *Browser*

Il *Browser* (WWW *Client*) è un componente abbastanza semplice all'interno dell'architettura presentata ed attraverso cui l'utente accede alle funzionalità offerte dal sistema; su di esso viene realizzata l'interfaccia grafica (*GUI-Graphical User Interface*) attraverso cui l'utente può interagire con il sistema. Le funzioni offerte dal *Browser* richiamano, attraverso il meccanismo *RMI-Remote Method Invocation*, metodi remoti realizzati sull' *Annotation Server*.

Il sistema è realizzato in modo tale da poter funzionare con *Browsers* diversi cercando di essere il più possibile *platform independent*.

7.1.5 Il *Proxy Server*

Il *Proxy Server* interagisce direttamente con i *Browsers* e con l'*Annotation Server* accedendo alle informazioni presenti sul *World Wide Web*. Sul *Proxy Server* viene realizzato il meccanismo di fusione tra le annotazioni ed i documenti a cui si riferiscono. Il *Proxy Server* richiama i metodi remoti messi a disposizione dall'*Annotation Server* al fine di soddisfare le richieste degli utenti del sistema. Il *Proxy Server* inoltre deve essere in grado di identificare la posizione dell'annotazione all'interno del documento evidenziando la porzione di testo a cui si riferisce con simboli grafici quali icone, sottolineature oppure barre luminose.

7.1.6 Modalità di comunicazione

La comunicazione tra i vari componenti è realizzata essenzialmente utilizzando il protocollo di rete TCP/IP ed il meccanismo RMI descritto nella sezione 6.5. La comunicazione tra lo strato applicativo presente sull'*Annotation Server* e la base di dati sottostante è realizzata tramite il meccanismo JDBC utilizzando *drivers Oracle* che offrono funzionalità specifiche e migliorano le prestazioni nell'utilizzo di un sistema Oracle per la gestione dei dati (sezione 6.4).

7.2 Meccanismo di autentificazione degli utenti

Il meccanismo di autentificazione degli utenti viene realizzato dall'*Annotation Server*; la scelta effettuata in fase di modellizzazione in riferimento al meccanismo di controllo degli accessi è quella relativa alla combinazione *userName + userPassword*.

TABLE KEYS

Key	Expires	IdProject	IdPerson
AS234SDF5567	07-11-1997	0	0
RT54645FGH5	07-11-1997	0	1
UIO546456GH	07-11-1997	1	1
345DRTFG556	07-11-1997	1	1
…	…	…	…

Tabella 7.1: Tabella delle chiavi d'accesso.

Ogni utente nel momento in cui viene registrato deve essere identificato attraverso un nome univoco ed una *password*; naturalmente di seguito l'utente ha la possibilità di modificare la propria *password* ogni volta che lo desidera. Inoltre ogni utente, effettuata la sua registrazione al

sistema, appartiene automaticamente al *Workspace Public* ed in seguito potrà essere aggiunto al gruppo di lavoro di un qualsiasi altro progetto[1].

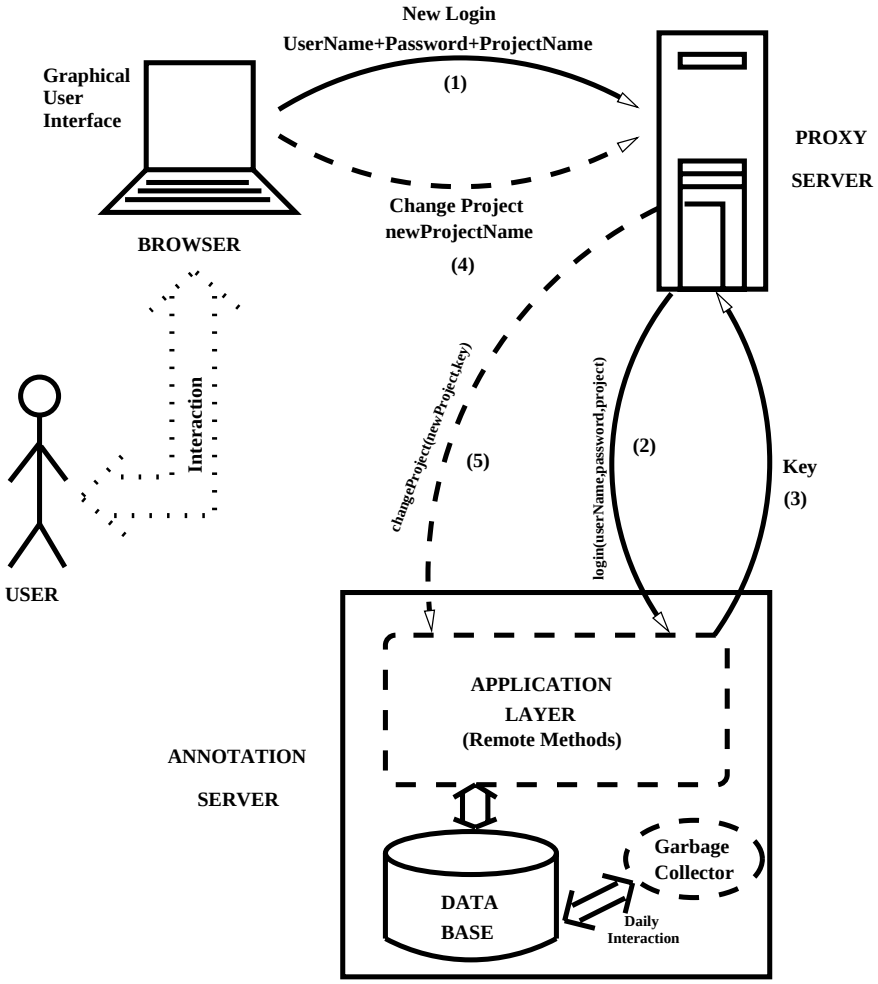

Figura 7.2: Meccanismo di autentificazione.

Internamente il sistema genera automaticamente una chiave casuale che identifica univocamente il *Login* effettuato da uno specifico utente nell'ambito di un determinato progetto. L'utente non ha accesso a tale chiave che è memorizzata in una tabella presente nella base di dati (tabella 7.1); essa deve essere utilizzata dal *Proxy Server* ogni volta che l'utente richiede l'invocazione di un metodo remoto presente sull'*Annotation Server*.

La tabella 7.1 mostra un esempio di come potrebbe essere costituita la tabella *KEYS* della base di dati; ogni tupla identifica un *Login* effettuato dall'utente. Il campo *Key* è la chiave alfanumerica casuale generata dall'*Annotation Server*, il campo *Expires* contiene la data in cui è stato effettuato il *Login* e viene utilizzato dal meccanismo di *Garbage Collection* descritto di seguito. I campi

[1]Tale operazione, come riportato nelle specifiche di progetto nel capitolo 5, può essere effettuata solo dal *Project Administrator*.

IdProject ed *IdPerson* contengono gli identificatori della persona e del progetto nell'ambito del quale è stato effettuato il *Login*.

La figura 7.2 mostra schematicamente la logica di funzionamento della procedura di autentificazione frazionandola in cinque *steps* numerati.

Nel momento in cui l'utente desidera accedere al sistema deve effettuare un *Login* specificando *userName, userPassword* e nome del progetto nell'ambito del quale intende lavorare (*step* (1)). A questo punto il *Proxy Server* invoca il metodo remoto realizzato sull'*Annotation Server* per effettuare il *Login* con il sistema (*step* (2)).

L'*Annotation Server*, fatti i controlli necessari a verificare la correttezza della *password* e l'appartenenza dell'utente al progetto specificato, determina automaticamente una chiave univoca che identifica il *Login* effettuato dall'utente (*step* (3)). Tale chiave viene restituita al *Proxy Server* e dovrà essere utilizzata in tutti i metodi remoti che la richiedono, come da specifiche riportate nella sezione 7.3.

Ogni utente potrà effettuare ovviamente diversi *Login* ognuno dei quali sarà identificato da un'univoca chiave d'accesso (tabella 7.1) in modo tale che il sistema possa ricostruire automaticamente l'ambiente (persona e progetto) a cui fa riferimento il *Login*.

L'utente inoltre ha la possibilità di modificare il progetto nell'ambito in cui lavora senza necessariamente fare un nuovo *Login* (*step* (4)). In tal caso l'*Annotation Server* non definisce una nuova chiave ma semplicemente modifica la tupla della tabella KEYS presente nella base di dati (tabella 7.1) con l'identificativo del nuovo progetto nell'ambito del quale l'utente intende lavorare.

Anche tale operazione viene realizzata da un metodo remoto che risiede sull'*Annotation Server* e che viene invocato dal *Proxy Server* (*step* (5)).

Nel momento in cui l'utente termina il proprio lavoro deve effettuare un *Logout* che cancella la tupla della tabella KEYS che si riferiva al *Login* effettuato. Nel caso in cui l'utente non effettui il *Logout* oppure sia impossibilitato ad effettuarlo, perchè ad esempio è caduta la connessione di rete, l'*Annotation Server* prevede un meccanismo di *Garbage Collection* della tabella KEYS.

Tale procedura è realizzata da un *Thread* attivato dall'applicazione *GarbageCollector.java* mandata in esecuzione nel momento in cui viene configurato l'*Annotation Server*.

Tale *Thread* entra automaticamente nello stato di esecuzione ogni giorno e cancella tutte le tuple della tabella KEYS che si riferiscono a *Login* effettuati in passato ed ancora aperti (attributo *Expires* della tabella 7.1); fatta la cancellazione il *Thread* torna a "dormire" per riattivarsi nuovamente il giorno successivo.

7.3 Classi ed interfacce dell'*Annotation Server*

In questa sezione è riportata una breve descrizione delle interfacce realizzate come oggetti remoti sull'*Annotation Server*.

L'implementazione di tali interfacce costituisce lo strato applicativo (*Application Layer*) realizzato sull'*Annotation Server* (vedi figura 7.1) e descritto nella sezione 7.1.3.

7.3.1 Introduzione

La figura 7.3 presenta una visione d'insieme delle interfacce applicative disponibili sull'*Annotation Server*, delle loro relazioni e delle classi che le implementano; la descrizione è stata effettuata utilizzando la sintassi UML (sezione 6.1).

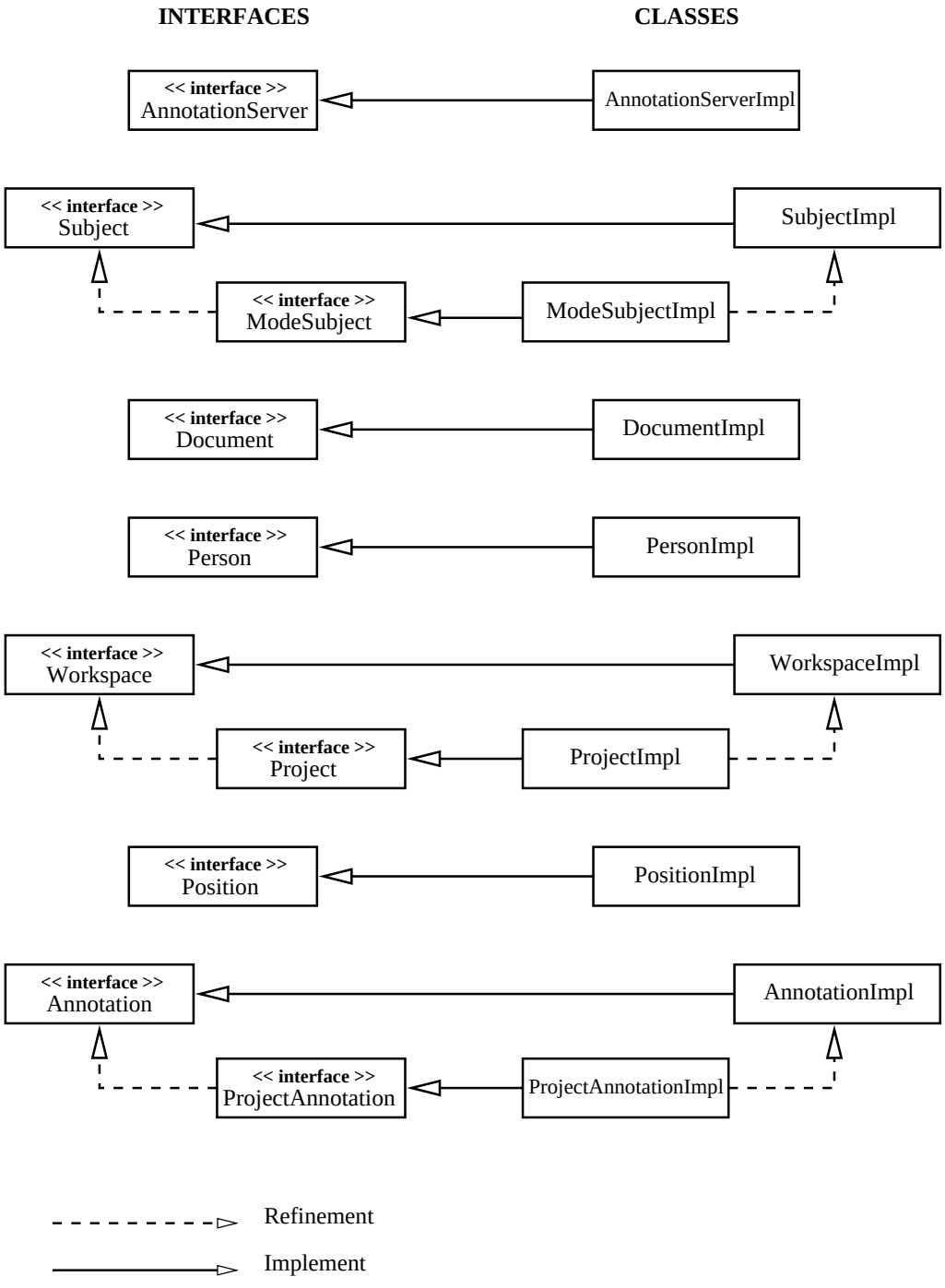

Figura 7.3: Classi ed interfacce dell'*Annotation Server*.

Ciascuna sottosezione riportata di seguito contiene la descrizione dei metodi realizzati per ogni oggetto remoto; il codice Java non è stato inserito in questo testo in quanto ciò avrebbe appesantito enormemente la trattazione costituendo, a mio avviso, un forte ostacolo alla comprensione dei vari argomenti esposti.

Prima di procedere con la descrizione di ciascun metodo è necessario fare alcune premesse di fondamentale importanza e riguardanti alcune caratteristiche generali possedute dai metodi elencati di seguito:

1. Tutti i metodi che richiedono il parametro di ingresso *String key* contengono il codice necessario a verificare l'identità dell'utente ed i sui diritti a svolgere le operazioni contemplate nel metodo stesso.

2. Alcuni metodi forniscono, come risultato della loro esecuzione, un parametro di tipo *Vector*. Il tipo *Vector* rappresenta una collezione di oggetti istanze di una qualsivoglia classe; un singolo vettore può contenere anche oggetti di classi diverse. Ogni metodo costruirà tale vettore in base alle specifiche funzionali secondo cui è stato implementato.

3. Tutti i metodi descritti sono realizzati in base al meccanismo *RMI Remote Method Invocation* (sezione 6.5), ovvero possono essere invocati remotamente da un qualsiasi *Browser Client* collegato con il sistema. Ogni metodo quindi si preoccupa di gestire anche le eccezioni legate al meccanismo di invocazione remota; le definizioni dei metodi riportate di seguito non presentano però tale aspetto al fine di rendere più chiara l'esposizione e la comprensione.

4. Ultimo aspetto generale che ha guidato la logica implementativa dei vari metodi riguarda la propagazione delle modifiche dallo strato applicativo alla base di dati (fare riferimento alla sezione 7.1). Sull'*Annotation Server* a livello applicativo gli oggetti che istanziano le varie classi riportate in figura 7.3 costituiscono un'immagine delle tabelle presenti nella base di dati (figura 5.20); la scelta effettuata è stata quella di garantire sempre la consistenza mantenendo costantemente aggiornato lo stato della base di dati. Tale scelta ha come svantaggio il calo di prestazioni che comunque è abbastanza contenuto dal momento che l'*Annotation Server* dispone di una base di dati locale (figura 7.1) a cui può accedere velocemente.

7.3.2 Classe *AnnotationServerImpl*

L'oggetto remoto *AnnotationServerImpl* è l'elemento fondamentale dell'*Annotation Server* che fa parte dell'architettura del sistema (sezione 7.1); esso mette a disposizione di un qualsivoglia *Browser* un insieme di servizi che consentono di realizzare le funzionalità del sistema di annotazione modellizzato (fare riferimento al capitolo 5). I metodi implementati nella classe *AnnotationServerImpl* sono riportati di seguito integrati da una breve spiegazione delle loro funzionalità.

È possibile raccogliere tali metodi in una serie di gruppi costituiti da elementi aventi le stesse caratteristiche funzionali e così classificati:

Gestione delle annotazioni: in questo gruppo sono raccolti tutti i metodi relativi alla gestione delle annotazioni, ovvero alla loro creazione, ricerca, visualizzazione e cancellazione.

Amministrazione degli utenti: tale insieme comprende tutti i metodi che consentono di amministrare l'insieme degli utenti registrati. A tal riguardo sono necessari metodi che consentano la registrazione degli utenti, la specifica dei profili che li caratterizzano, l'aggiunta e la cancellazione degli utenti ai vari progetti, la possibilità di effettuare *Login* e *Logout* con il sistema e la loro cancellazione dal sistema.

Amministrazione dei progetti: in questo gruppo sono compresi tutti i metodi che consentono di svolgere tutti i compiti necessari all'amministrazione dei progetti gestiti dal sistema. A tal riguardo occorre ricordare che i progetti sono organizzati gerarchicamente come estesamente descritto nella sezione 5.5.3.

Amministrazione dei documenti: tali metodi sono relativi alla gestione dei documenti a cui si riferiscono le annotazioni. Il sistema tiene traccia di tali documenti per mezzo del loro URL in base al quale possono essere univocamente identificati e reperiti dal *World Wide Web*.

Amministrazione dei soggetti: l'*Annotation Server* deve fornire anche un insieme di metodi che gestiscano i soggetti (temi di interesse) che possono essere associati ad annotazioni, progetti ed utenti.

Notificazione: questo gruppo è costituito da un solo metodo che si occupa di realizzare il meccanismo di notificazione tramite *email* descritto nella sezione 5.8.2.

Ultima considerazione generale è che la classe *AnnotationServerImpl* non presenta alcun attributo ma mette a disposizione solo dei servizi.

Gestione delle annotazioni

- **Vector getAllAnnotations ();**

  Questo metodo restituisce un vettore contenente tutte le annotazioni che sono memorizzate nella base di dati senza applicare alcun filtro e senza effettuare alcuna particolare ricerca.

- **Annotation getAnnotationById(long id);**

  Tale metodo restituisce un oggetto della classe *AnnotationImpl* avente l'identificatore specificato come parametro. L'identificatore fa riferimento alla base di dati.

- **ProjectAnnotation getProjectAnnotationById(long id);**

  Tale metodo restituisce un oggetto della classe *ProjectAnnotationImpl* avente l'identificatore specificato come parametro.

- **Vector findAnnotationsOfDocument (String url, String key);**

  Con tale metodo l'utente registrato nella base di dati può ottenere tutte le annotazioni che ha il diritto di leggere e che si riferiscono al documento identificato per mezzo del suo URL passato come parametro. I diritti vengono determinati in base al *login* effettuato (sezione 7.2) e per questa ragione il metodo richiede un secondo parametro che è la chiave che viene assegnata all'utente nel momento in cui effettua il *login* con il sistema specificando anche il progetto in cui intende lavorare.

- **Vector findAnnotationsOfAnnotation (long idAnnotation, String key);**

  Questo metodo è analogo al precedente ma consente di ottenere tutte le annotazioni che si riferiscono ad una determinata annotazione specificata come parametro.

- **Vector findAnnotationsOfPerson (String name, String key);**

  Tale metodo costruisce un vettore contenente tutte le annotazioni fatte da una determinata persona nell'ambito del progetto in cui si sta lavorando od in uno dei suoi superprogetti. Le *Private Annotations* vengono selezionate solo se la persona di cui si richiedono le annotazioni è l'autore dell'annotazione.

- **Vector findAnnotationsOfSubject (String name, String key);**

  Tale metodo costruisce un vettore contenente tutte le annotazioni relative ad un determinato soggetto che l'utente ha il diritto di leggere in base al *login* che ha effettuato con il sistema.

- **Vector findAnnotationsOfProject(String name, String key);**

  Questo metodo offre la possibilità di ottenere dalla base di dati tutte le annotazioni che si riferiscono ad un determinato progetto. Anche in tale metodo i diritti[2] sulle annotazioni vengono determinati in base al *login* effettuato dall'utente ovvero in base al suo identificatore ed al progetto in cui sta lavorando.

- **ProjectAnnotation addAnnotation (Vector subjects, String title, String day, String text, String url, Position position, String relevants, String key);**

  Tale versione del metodo *addAnnotation* permette l'aggiunta alla base di dati di una nuova *ProjectAnnotation* che fa riferimento ad uno specifico documento. Nell'implementazione del metodo si sono valutati tutti i possibili casi di annotazioni composte in modo errato; in tal caso la *Project Annotation* fornita all'esterno da tale metodo conterrà dei valori di default e del tutto privi di significato. Tale metodo inoltre realizza anche il meccanismo di notificazione attraverso la spedizione di *emails* rendendo l'*Annotation Server* un componente attivo all'interno dell'architettura del sistema.

- **ProjectAnnotation addAnnotation (Vector subjects, String title, String day, String text, long parent, Position position, String relevants, String key);**

[2]Con il termine "diritti" si fa riferimento all'insieme di annotazioni che possono essere lette e quindi annotate dall'utente che sta effettuando la ricerca.

Questa versione del metodo *addAnnotation* consente la composizione di un'annotazione di tipo *Project* che si riferisce ad un'altra annotazione; anche tale versione comprende tutti i controlli necessari affinché l'annotazione composta rispetti i requisiti richiesti per la sua memorizzazione nella base di dati. Tutte le versioni del metodo *addAnnotation* determinano automaticamente l'identificatore univoco con il quale le annotazioni vengono memorizzate nella base di dati.

- **Annotation addAnnotation (Vector subjects, String title, String day, String text, String url, Position position, String key);**

  Tale metodo consente la composizione di una *Private Annotation* riferita ad un determinato documento; in tal caso non è necessario alcun meccanismo di notificazione in quanto nessuno, se non il proprietario, potrà leggere un'annotazione privata. Questo metodo consente quindi la realizzazione del principio di **personalizzazione** a lungo discusso nel corso di tale testo.

- **Annotation addAnnotation (Vector subjects, String title, String day, String text, long parent, Position position, String key);**

  Tale metodo consente la creazione di una nuova *Private Annotation* che si riferisce ad un'altra annotazione; anche l'implementazione di tale versione del metodo comprende tutti i controlli necessari affinché l'annotazione rispetti i requisiti rischiesti ed esposti nel capitolo 5.

- **boolean deleteAnnotation (Annotation annotation, String key);**

  Tale metodo offre la possibilità di cancellare le annotazioni solo all'amministratore del sistema. A seguito della cancellazione viene innescato il meccanismo di notificazione così come specificato nella sezione 5.8.3. Se l'annotazione selezionata per la cancellazione presenta una gerarchia sottostante, ovvero altre annotazioni che si riferiscono ad essa, esse vengono eliminate automaticamente dall'*Annotation Server* con i conseguenti messaggi di notifica.

Amministrazione degli utenti

- **Person getPerson(String key);**

  Con tale metodo è possibile ottenere un oggetto della classe *PersonImpl* contenente tutte le informazioni relative all'utente che ha effettuato il *login*.

- **Person getPersonByName(String name);**

  Questo metodo fornisce un oggetto della classe *PersonImpl* referenziato attraverso il nome univoco passato come parametro.

- **Person addUser (String name, String password, String email);**

Con questo metodo è possibile aggiungere un utente alla base di dati. L'utente viene automaticamente aggiunto tra i membri del *Workspace Public* ed ad egli viene riferito il soggetto *Public*.

- **boolean addUserToProject (String name, String key);**

  Tale metodo consente all'amministratore di un determinato progetto di aggiungere dei membri al gruppo di persone che già vi lavorano. L'utente deve essere già registrato nella base di dati per poter essere aggiunto al gruppo di progetto altrimenti l'esecuzione del metodo non avrà alcun effetto.

- **boolean deleteUser (String name, String key);**

  Con questo metodo l'amministratore del sistema può cancellare un utente dalla base di dati. Il metodo inoltre effettua automaticamente tutti i controlli necessari per verificare che l'utente non sia referenziato in istanze di altre tabelle della base di dati. In caso contrario la cancellazione non viene permessa.

- **boolean deleteUserFromProject (String name, String key);**

  Questo metodo offre la possibilità al *Project Administrator* di eliminare dei membri dal gruppo di progetto. A nessuno, nemmeno al *System Administrator*, è permessa la cancellazione dell'amministratore del sistema dal *Workspace Public* perché ciò fa parte della configurazione del sistema[3] stesso. Inoltre il *Project Administrator* non può cancellare un utente che sta lavorando al progetto ovvero che ha una sessione aperta su un determinato *Browser* in quanto ha in precedenza effettuato un *login*. Infine gli utenti possono essere cancellati dal *Workspace Public* solo se non appartengono a nessuno dei progetti facenti parte dell'albero gerarchico. In caso contrario occorre prima cancellare gli utenti da tutti i progetti di cui fanno parte e quindi eliminarli dal *Workspace Public*.

- **String login (String user, String password, String project);**

  Questo metodo effettua il *login* dell'utente generando una chiave d'accesso in modo casuale ed univoco. Il codice di tale metodo effettua tutti i controlli necessari a verificare se la *password* fornita è corretta e se l'utente appartiene effettivamente al progetto in cui desidera lavorare. Fatto ciò viene generata la chiave casuale e restituita al *Proxy Server* come una stringa di caratteri, che dovrà essere utilizzata in tutti gli altri metodi che richiedono la procedura di autentificazione.

- **void logout(String key);**

  Questo metodo effettua il *logout* dell'utente a cui appartiene la chiave passata come parametro; il *logout* consiste semplicemente nella cancellazione della tupla appartenente alla tabella KEYS ed avente come chiave d'accesso il parametro in ingresso a tale metodo.

[3] Fare riferimento alla tabella 8.1.

- **boolean changeProject (String newProject, String key);**

  Tale metodo viene utilizzato dall'utente nel momento in cui ha già effettuato il *login* e desidera cambiare il progetto in cui sta lavorando passando ad un altro di cui però deve sempre esserne membro; in tal caso la chiave d'accesso resta sempre la stessa mentre viene modificato l'identificativo del processo a cui si riferisce.

Amministrazione dei progetti

- **Vector getAllProjectsOfPerson(String key);**

  Questo metodo fornisce un vettore contenente tutti i progetti, indipendentemente dalla loro collocazione nell'albero gerarchico, a cui appartiene l'utente che ha effettuato il *login*.

- **Project getProjectByName(String name);**

  Questo metodo fornisce un oggetto di classe *ProjectImpl* sulla base del suo nome passato come parametro. Ciò è possibile in quanto il nome del progetto è univoco tra tutti quelli registrati nella base di dati.

- **Project getPublic();**

  Questo metodo fornisce un oggetto della classe *ProjectImpl* corrispondente al *Workspace Public*.

- **Vector getProjectsOfPerson(String key);**

  Questo metodo fornisce un vettore contenente oggetti della classe *ProjectImpl* e rappresentanti tutti i progetti che si trovano nella gerarchia registrata nel sistema, percorsa dal basso verso l'alto a partire dal progetto in cui sta lavorando attualmente l'utente che richiama tale metodo.

- **Project addProject (String nameProject, String description, String nameAdministrator, String password, String email, String timespan, String goals, String nameSubject, String key);**

  Questo metodo consente la creazione di un nuovo progetto con un nome univoco all'interno dell'albero gerarchico mantenuto dal sistema. La base di dati viene aggiornata automaticamente con tutte le informazioni necessarie.

- **int deleteProject (Project project, String key);**

  Questo metodo offre la possibilità al *Project Administrator* di cancellare il progetto da lui amministrato. La cancellazione è resa possibile solo se il progetto non ha figli. Il parametro ritornato da tale metodo fornisce indicazioni circa l'esito del metodo stesso e, in caso di esito negativo, i valori assunti da tale parametro sono diversi a seconda delle ragioni del fallimento; a tal riguardo le specifiche dettagliate sono riportate nel codice che implementa l'interfaccia *AnnotationServer*.

Amministrazione dei documenti

- **Vector getDocuments();**

  Questo metodo fornisce la lista di tutti i documenti presenti nella base di dati.

- **Document addDocument (String url);**

  Questo metodo da la possibilità di aggiungere un determinato documento alla base di dati. Nel caso in cui il documento è già presente nella base di dati il sistema non effettua alcun aggiornamento.

- **boolean deleteDocument(String url);**

  Questo metodo da la possibilità ad ogni utente registrato nel sistema di cancellare un documento; ciò è reso possibile solo se non vi è nessuna annotazione che si riferisce al documento in questione. Ovviamente l'operazione di cancellazione è effettuata dal sistema di annotazione, nel senso che viene cancellato il riferimento al documento interno al sistema, non certo il documento originale dal *Server WWW* dove si trova e dove continua ovviamente ad esistere.

Amministrazione dei soggetti

- **Vector getSubjects();**

  Questo metodo fornisce la lista di tutti i soggetti presenti nella base di dati.

- **Subject addSubject (String name);**

  Questo metodo consente la creazione di un nuovo soggetto se questo non è già presente nella base di dati.

- **boolean deleteSubject(String name);**

  Questo metodo viene utilizzato nel momento in cui si desidera cancellare un determinato *Subject* dalla base di dati; il sistema non consente la cancellazione se vi è almeno un utente, un progetto od un'annotazione che si riferiscono al soggetto che si desidera eliminare.

Notificazione

- **public void sendEmailToChildren(Vector children, Connection conn, long id, String emailAdministrator, String title, String url, String relevants, String day);**

  Tale metodo viene utilizzato implicitamente dai metodi *addAnnotation* per realizzare il meccanismo di notificazione attraverso la composizione e la spedizione di una *email* agli utenti determinati in base ai principi esposti nella sezione 5.8.2.

7.3.3 Classe *AnnotationImpl*

La classe *AnnotationImpl* realizza a livello applicativo l'immagine di una *Private Annotation* memorizzata nella base di dati. Tutti gli attributi di tale classe sono definiti *Private*[4] e sono descritti nella sezione 5.5.5. Tali metodi non sono stati integrati da una spiegazione in quanto il loro nome è autoesplicativo.

- **long getId();**
- **Vector getSubjects();**
- **String getTitle();**
- **String getDay();**
- **Person getAuthor();**
- **String getText();**
- **int getTypology();**
- **Document getDocument();**
- **long getParent();**
- **Position getPosition();**
- **Workspace getBelongs();**

7.3.4 Classe *ProjectAnnotationImpl*

La classe *ProjectAnnotationImpl* estende la classe *AnnotationImpl* con il solo attributo *relevants* realizzando così a livello applicativo l'immagine di una *Project Annotation*. L'attributo *relevants* è un oggetto della classe *WorkspaceImpl* ed indica il progetto nell'ambito del quale l'annotazione è rilevante; di seguito è riportato il metodo per la gestione di tale attributo.

- **Workspace getRelevants() throws java.rmi.RemoteException;**

  Tale metodo fornisce lo spazio di lavoro nell'ambito del quale l'annotazione è rilevante.

[4] Ciò significa che solo la classe stessa può utilizzarli in termini sia di lettura che di scrittura del loro contenuto.

7.3.5 Classe *WorkspaceImpl*

La classe *WorkspaceImpl* è costituita da un insieme di metodi e di attributi che costituiscono, a livello applicativo, un generico "spazio di lavoro" che può essere il *Workspace Public* oppure un progetto. Gli attributi di tale classe sono stati dichiarati *Private* mentre i metodi sono riportati di seguito. I metodi che presentano il parametro *Key* in ingresso effettuano anche i controlli necessari per verificare l'identità della persona che sta effettuando l'operazione. La logica del meccanismo di autentificazione è spiegata estesamente nella sezione 7.2.

Occorre ricordare che i progetti sono organizzati gerarchicamente ed identificabili per mezzo di un nome univoco costruito utilizzando la *dot notation*; tale nome consente quindi di ottenere l'informazione circa la posizione del progetto nell'albero gerarchico.

- **long getId();**

  Questo metodo consente di ottenere l'identificatore dell'oggetto della classe *WorkspaceImpl*.

- **String getName();**

  Questo metodo restituisce il nome completo del *Workspace* in riferimento al quale viene chiamato. Per nome completo si intende il nome esteso rappresentato per mezzo della *dot notation*, ovvero in funzione della posizione del *Workspace* nell'albero gerarchico dei progetti.

- **Person getAdministrator();**

  Questo metodo fornisce un oggetto di tipo *PersonImpl* che contiene tutte le informazioni relative all'amministratore del progetto.

- **Vector getMembers();**

  Questo metodo fornisce all'esterno un vettore costituito da oggetti della classe *PersonImpl* e che rappresentano i membri del progetto. Anche l'amministratore, sebbene non identificabile in una persona fisica, è un membro del progetto e come tale è un elemento del vettore restituito dal metodo *getMembers()*.

- **boolean setName(String name, String key);**

  Questo metodo consente la definizione del nome di uno specifico *Workspace*; il nome passato come parametro è quello sintetico che si riferisce allo spazio di lavoro. Il metodo *setName(String name,String key)* costruisce automaticamente il nome completo utilizzando la *dot notation*[5] e lo memorizza quindi nella base di dati. Inoltre tale metodo modifica automaticamente i nomi di tutti i progetti che si trovano nella gerarchia sottostante al progetto a cui si è cambiato il nome. Ciò è necessario in quanto i nomi dei progetti sono costruiti con la notazione puntata e quindi dipendono dalla loro posizione nell'albero gerarchico.

[5] Utilizzando tale notazione il nome rispecchia completamente la gerarchia soprastante al progetto in questione.

- **boolean setAdministrator(String name, String key);**

  Questo metodo consente la definizione di un nuovo amministratore di progetto. Solo il vecchio amministratore può effettuare tale operazione, previa registrazione del nuovo *Project Administrator* nel sistema.

- **boolean addMember(String name, String key);**

  Questo metodo può essere utilizzato per aggiungere nuovi componenti al proprio gruppo di progetto. Il *System Administrator* non può fare parte di nessun gruppo di progetto se non del gruppo del *Workspace Public*. Un nuovo membro può essere aggiunto solo se precedentemente registrato nel sistema utilizzando il metodo apposito riportato nella classe *AnnotationServerImpl*.

- **boolean deleteMember(String name, String key);**

  Questo metodo consente la cancellazione di un membro da un determinato gruppo di progetto; i membri cancellati dal gruppo di progetto non vengono cancellati automaticamente dalla base di dati in quanto essi potrebbero fare parte di altri progetti. Inoltre con tale metodo non può essere cancellato l'amministratore di progetto perché ciò potrebbe portare ad inconsistenza nei dati ovvero alla possibile presenza di progetti senza amministratore; ma questa situazione non è prevista dalle specifiche di progetto. Non possono inoltre essere cancellati gli utenti che hanno un *login* aperto nell'ambito del progetto da cui l'amministratore li vuole eliminare. In tal caso occorre che l'utente effettui un *logout* e quindi procedere con la cancellazione.

7.3.6 Classe *ProjectImpl*

La classe *ProjectImpl* rappresenta, a livello applicativo, un generico progetto, che qualsiasi utente registrato nel sistema può definire. Tale classe estende la classe *WorkspaceImpl* che invece rappresenta un generico spazio di lavoro tra cui anche il *Workspace Public*. Ogni oggetto della classe *ProjectImpl* possiede un superprogetto a cui si riferisce, una descrizione, un intervallo di tempo entro cui deve essere portato a termine e degli obiettivi che deve perseguire. Di seguito sono riportati i metodi che tale classe mette a disposizione.

- **Workspace getParent();**

  Questo metodo fornisce il progetto che si trova nella posizione direttamente soprastante rispetto a quella del progetto corrente nella gerarchia dei progetti registrata nel sistema.

- **Vector getSubjects();**

  Tale metodo fornisce un vettore contenente i soggetti del progetto corrente; gli oggetti di tale vettore fanno parte della classe *SubjectImpl*.

- **boolean addSubject(String name, String key;**

Questo metodo offre la possibilità di aggiungere un nuovo soggetto ai soggetti a cui si riferisce un determinato progetto; solo il *Project Administrator* può svolgere tale operazione. Questo metodo inoltre aggiorna anche la base di dati.

- **boolean deleteSubject(String name, String key);**

  Con tale metodo il *Project Administrator* può cancellare un soggetto che si riferisce al progetto da lui amministrato. La base di dati viene aggiornata ma il soggetto non viene cancellato dalla tabella SUBJECT perché esso potrebbe essere legato ad altri progetti, annotazioni od utenti.

- **Vector getChildren();**

  Utilizzando tale metodo è possibile ottenere un vettore contenente tutti i progetti che si trovano al di sotto di un determinato progetto nella gerarchia registrata nel sistema. Ogni elemento del vettore fornito all'esterno è un oggetto della classe *ProjectImpl*. Tale metodo accede alla base di dati per ottenere tutti i dati necessari relativi ai sottoprogetti.

- **String getDescription();**

  Questo metodo fornisce la descrizione generica del progetto corrente.

- **String getTimespan();**

  Con tale metodo è possibile conoscere quale è la durata prevista del progetto sul quale è richiamato.

- **String getGoals();**

  Questo metodo fornisce all'esterno una stringa di caratteri rappresentante gli obiettivi di un determinato progetto.

- **boolean setDescription(String description, String key);**

  Con tale metodo il *Project Administrator* può modificare in maniera permanente, ovvero aggiornando la base di dati, il campo *description* del progetto corrente.

- **boolean setTimespan(String timespan, String key);**

  Con tale metodo il *Project Administrator* può modificare in maniera permanente, ovvero aggiornando la base di dati, il campo *timespan* del progetto corrente.

- **boolean setGoals(String goals, String key);**

  Con tale metodo il *Project Administrator* può modificare in maniera permanente, ovvero aggiornando la base di dati, il campo *goals* del progetto corrente.

7.3.7 Classe *PersonImpl*

La classe *PersonImpl* viene utilizzata, a livello delle applicazioni residenti sull'*Annotation Server*, per la gestione della tabella PERSON (figura 5.20) implementata nella base di dati sottostante. I metodi remoti che possono essere richiamati da un qualsiasi *Browser* sono elencati di seguito con una breve descrizione delle loro funzionalità. Tutti gli attributi della classe *PersonImpl* sono dichiarati *Private* compreso il vettore dei soggetti che rappresentano i campi di ricerca a cui una determinata persona è interessata.

- **long getId();**

  Questo metodo restituisce l'identificatore utilizzato nella base di dati per referenziare univocamente l'oggetto della classe *PersonImpl*.

- **String getName();**

  Questo metodo restituisce il nome univoco della persona in questione.

- **boolean setName(String name, String key);**

  Questo metodo consente all'utente di modificare il proprio nome registrato nella base di dati. Il meccanismo di autentificazione è realizzato per mezzo del parametro *Key* passato come argomento del metodo. Il valore del parametro in uscita al metodo è *true* se la modifica è andata a buon fine oppure *false* se l'autentificazione non ha avuto successo.

- **boolean setPassword(String password, String Key);**

  Questo metodo offre all'utente la possibilità di modificare la propria *password*; la semantica è analoga a quella del metodo precedente.

- **String getEmail();**

  Questo metodo fornisce l'*email* di una determinata persona

- **boolean setEmail(String email, String key);**

  Questo metodo consente la modifica dell'*email* da parte dell'utente. La logica di funzionamento è analoga a quella dei metodi utilizzati per la modifica della *password* o del nome.

- **Vector getSubjects();**

  Questo metodo restituisce un vettore contenente i *Subjects* a cui è interessata una determinata persona. Ogni elemento del vettore è un'istanza della classe *ModeSubjectImpl*.

- **boolean addSubject(String name, String key, int mod);**

  Questo metodo consente all'utente di specificare un proprio soggetto di interesse definendone la modalità in base alla quale è interessato. Nel caso in cui il soggetto non è presente nella base di dati tale metodo lo crea automaticamente. Nel caso in cui l'utente sia già

interessato al soggetto, ma in base ad una modalità diversa, il metodo cambia solamente la modalità di interesse. Il metodo ovviamente prevede la presenza del meccanismo di autentificazione in modo tale che ogni utente può definire solamente i propri soggetti di interesse e non modificare quelli di altri utenti.

- **boolean deleteSubject(String name, String key);**

  Questo metodo consente la cancellazione del legame tra la persona ed un determinato soggetto. Con tale metodo non viene effettuata la cancellazione del soggetto dalla base di dati; per fare ciò occorre utilizzare lo specifico metodo definito nella classe *AnnotationServerImpl*.

- **boolean deleteSubject(String name, String userName, String administratorKey);**

  Tale metodo offre la possibilità all'amministratore del sistema di cancellare esplicitamente il legame tra un determinato soggetto ed uno specifico utente registrato nella base di dati. Questo è un metodo fondamentalmente a carattere amministrativo in quanto il sistema non offre la possibilità di cancellare un utente se quest'ultimo possiede ancora dei soggetti di interesse. L'amministratore del sistema, utilizzando tale metodo, potrà cancellare eventuali collegamenti tra utente e soggetti e quindi procedere all'eliminazione dell'utente stesso.

7.3.8 Classe *DocumentImpl*

L'oggetto remoto *DocumentImpl* è stato pensato al fine di avere, a livello applicativo, l'immagine della tabella DOCUMENT (figura 5.20) implementata nella base di dati. Tale classe presenta degli attributi che sono stati definiti come *Private* e che sono resi disponibili all'esterno per mezzo dei metodi sotto riportati:

- **long getId();**

  Questo metodo fornisce l'identificatore numerico ed univoco del documento in oggetto e che si trova memorizzato nel *Database*.

- **String getURL();**

  Questo metodo fornisce l'URL del documento in questione; anche l'URL deve essere univoco ed è costituito da una stringa di caratteri.

7.3.9 Classe *SubjectImpl*

La classe *SubjectImpl* da la possibilità a livello applicativo di avere un oggetto contenente un'istanza della tabella SUBJECT realizzata nella base di dati. Tale classe è molto semplice in quanto possiede solamente l'attributo *name* che identifica univocamente ciascun soggetto.

- **String getName();**

  Questo metodo da la possibilità di ottenere l'attributo *name* del soggetto che costituisce l'istanza della classe *SubjectImpl*.

Non sono previsti metodi per la modifica del nome del soggetto; se si vuole fare ciò si deve prima cancellare il soggetto esistente e quindi inserire il nuovo soggetto utilizzando i metodi forniti dalla classe *AnnotationServerImpl*.

La ragione che giustifica tale scelta consiste essenzialmente nel fatto che la modifica del nome di un soggetto potrebbe portare ad inconsistenza nei dati memorizzati nel sistema, con particolare riferimento alle annotazioni create in passato che potrebbero avere un soggetto il cui nome è stato modificato e quindi non esiste più nella base di dati. In tal caso il meccanismo di propagazione automatica degli effetti non è logicamente corretto in quanto, ad esempio, il nuovo nome del soggetto potrebbe non essere consistente con il contenuto dell'annotazione.

7.3.10 Classe *ModeSubjectImpl*

La classe *ModeSubjectImpl* è una sottoclasse di *SubjectImpl* e quindi ne eredita i metodi e gli attributi. Inoltre la classe *ModeSubjectImpl* presenta l'attributo numerico *mode* attraverso il quale l'utente può specificare la modalità in base alla quale è interessato ad un determinato soggetto. Se l'utente è interessato in modalità PUBLIC ad un soggetto l'attributo *mode* assume il valore 0 altrimenti, nel caso in cui l'utente è interessato in modalità NOPUBLIC ad un determinato soggetto, l'attributo *mode* vale 1.

Attraverso l'attributo *mode* l'utente può quindi definire esplicitamente la modalità in base alla quale essere interessato ad un determinato soggetto limitando il numero di notifiche ricevute; questo attributo è quindi strettamente correlato al meccanismo di notificazione esposto nella sezione 5.8.2.

- **int getMode();**

  Questo metodo rende disponibile l'attributo *mode* della particolare istanza della classe *ModeSubjectImpl*.

- **boolean setMode(int mode, String key);**

  Questo metodo da la possibilità all'utente interessato ad un determinato soggetto di definire ed eventualmente modificare la modalità in base alla quale è interessato ad un determinato soggetto. Il parametro *key* consente l'identificazione univoca dell'utente realizzando il meccanismo di autentificazione spiegato nella sezione 7.2.

7.3.11 Classe *PositionImpl*

La classe *PositionImpl* rappresenta la posizione a cui può essere riferita l'annotazione all'interno di un determinato documento o di un'altra annotazione. Il servizo offerto da tale classe è riportato di seguito.

- **String getPosition();**

  Tale metodo fornisce una stringa di caratteri rappresentante la posizione dell'annotazione.

Tale classe non prevede metodi per la modifica della posizione di un'annotazione all'interno del documento; tale operazione non è resa possibile dal sistema coerentemente ai principi definiti in fase di analisi e riportati nel capitolo 5. La posizione viene definita dall'utente al momento della creazione di un'annotazione utilizzando i metodi forniti dalla classe *AnnotationServerImpl* e poi non può più essere modificata.

7.4 Gestione degli errori

Il meccanismo di gestione degli errori è realizzato prevalentemente sull'*Annotation Server* nell'ambito di ogni metodo remoto che può essere richiamato dal *Proxy Server*. Prima di effettuare ogni operazione sulla base di dati, l'*Annotation Server* controlla l'identità dell'utente e quindi identifica eventuali violazioni di diritti, vincoli di integrità, unicità dei valori e così via. Nel caso in cui i controlli forniscono un esito negativo non viene eseguita alcuna operazione sulla base di dati.

In aggiunta a tali meccanismi di gestione degli errori realizzati attraverso implementazione di codice ad hoc, si sono utilizzate anche le procedure per sollevare eccezioni messe a disposizione dai *packages* del linguaggio Java. Inoltre l'interfaccia utente realizzata sul *Browser* è stata concepita in modo tale da limitare la possibilità che l'utente richiami metodi con parametri errati o che violino determinati vincoli definiti in fase di modellizzazione. Ciò ha reso necessaria la realizzazione di metodi specifici offerti dall'*Annotation Server* ed aventi lo scopo di guidare l'utente in maniera *User Friendly* nell'utilizzo del sistema con un notevole risparmio di tempo e di energie.

La base di dati è stata progettata facendo uso dei vincoli di integrità referenziale che si possono specificare facendo uso del *Data Definition Language* facente parte del linguaggio SQL (sezione 6.2.1). In questo modo vengono gestiti automaticamente tutti i possibili errori relativi ad inconsistenza di dati a seguito di operazioni compiute dall'utente; questi errori vengono propagati verso l'alto in modo tale che l'utente ne possa venire a conoscenza.

Ovviamente tale meccanismo non prevede una propagazione diretta dei messaggi d'errore sollevati dal DBMS bensì una loro manipolazione e gestione a livello applicativo. Ciò rispetta il principio di completa trasparenza a livello di interfaccia utente rispetto a quanto accade al livello sottostante di base di dati.

Parte III

Annotation Server: implementazione, applicazioni e valutazioni

Capitolo 8

Implementazione ed applicazioni

Questo capitolo presenta una descrizione di alcuni dettagli implementativi ritenuti particolarmente significativi nonché una breve rassegna circa i possibili campi d'impiego del sistema per l'annotazione di documenti multimediali presentato in tale testo.

La prima parte del capitolo è dedicata alla descrizione dell'implementazione della base di dati, dei meccanismi di interazione tra la base di dati e lo strato applicativo realizzato sull'*Annotation Server* e dei meccanismi RMI (*Remote Method Invocation*) per l'invocazione di metodi remoti.

Segue una sezione dedicata alla descrizione di alcuni esempi che mostrano le modalità di realizzazione del sistema facendo uso di una base di dati attiva, attraverso cui è possibile implementare regole attive che rispecchiano i meccanismi analizzati e classificati nella sezione 5.9. Scopo di tali esempi è quello di mostrare come è possibile realizzare un prototipo funzionante sulla base di regole attive integrate nell'implementazione della base di dati.

Nella parte conclusiva verranno illustrati i possibili campi di impiego del sistema di annotazione proposto. In particolare si farà accenno al progetto WEL (*Warburg Electronic Library*, sezione 4.2.2) nell'ambito del quale verrà impiegato il sistema, ponendo particolare enfasi sugli aspetti di personalizzazione e di cooperazione che hanno costituito un riferimento essenziale nel corso di tutta la progettazione e sviluppo del sistema.

Al fine di non appesantire la trattazione dei vari argomenti si è deciso di riportare nell'appendice C solo alcuni esempi tratti dall'intero codice realizzato; l'analisi di tali esempi costituisce un presupposto fondamentale per meglio comprendere il contenuto di questo capitolo.

8.1 Dettagli implementativi

Questa sezione si struttura in tre parti ben distinte, ciascuna delle quali riporta alcuni dettagli relativi ai diversi aspetti affrontati in fase di implementazione del prototipo. La scelta effettuata è stata quella di riportare il codice realizzato solo laddove strettamente necessario ai fini della comprensione dell'argomento trattato.

Questa sezione si configura quindi particolarmente tecnica e non è necessaria al fine del-

la comprensione dei concetti principali esposti nel corso del testo, quanto invece per avere informazioni dettagliate circa alcune scelte effettuate in fase di implementazione.

8.1.1 Implementazione del *Database*

L'implementazione del *Database* è stata effettuata sulla base dello schema logico riportato nel capitolo 5; è stata realizzata un'applicazione Java per la creazione delle tabelle e l'inizializzazione del sistema attraverso l'inserimento del *Workspace Public* e dei dati relativi al *System Administrator*; inoltre è stata creata un'applicazione Java per la cancellazione delle tabelle che, normalmente, non verrà mai utilizzata in quanto il *Database* deve essere sempre attivo. Il DBMS utilizzato è *Oracle v8.0* ed il linguaggio di interrogazione della base di dati è SQL.

Progettazione fisica

In questa fase lo schema logico riportato nella sezione 5.6.2 viene completato con la specifica dei parametri fisici di memorizzazione dei dati. Il prodotto di tale fase viene chiamato schema fisico. Tale schema fa riferimento al modello fisico dei dati, che dipende dallo specifico sistema di gestione della base di dati scelto e che tiene conto dei criteri di organizzazione fisica dei dati nel sistema sviluppato (nel prototipo realizzato è stato utilizzato il DBMS *Oracle v.8.0*).

Nella figura 8.1 è riportata, a titolo di esempio, una porzione di codice dell'applicazione che crea le tabelle[1] con le specifiche del tipo di ogni attributo identificato nello schema concettuale (sezione 5.6.1).

```
class CreateTables {
  public static void main (String args[]){
    try{
..............
//create table DOCUMENT
stmt.executeUpdate("CREATE TABLE DOCUMENT(id NUMBER(10),
url VARCHAR(200) NOT NULL UNIQUE, PRIMARY KEY (id),
CHECK (id>=0))");
System.out.println("Table DOCUMENT created");
..............
```

Figura 8.1: Creazione della tabella DOCUMENT.

Come si può facilmente dedurre dal codice di figura 8.1 la tabella DOCUMENT possiede solo due attributi (un identificatore numerico e l'URL del documento memorizzato sotto forma di stringa di caratteri); tramite l'istruzione *stmt.executeUpdate* è possibile utilizzare il meccanismo

[1] L'esempio riportato è relativo alla tabella DOCUMENT che è molto semplice ed è costituita da pochi attributi. Lo scopo è quello di spiegare il meccanismo di creazione senza riportare lunghe porzioni di codice che appesantirebbero notevolmente l'esposizione.

JDBC per sottoporre il comando *CREATE TABLE* alla base di dati e creare in tal modo la tabella desiderata.

8.1.2 L'interazione tra lo strato applicativo ed il *Database*

In tale sezione verranno presentati in dettaglio alcuni aspetti dell'implementazione relativi al meccanismo di interazione tra le applicazioni di gestione del *Database* scritte in linguaggio Java e la base di dati relazionale sottostante[2] la cui implementazione è presentata nella sezione 8.1.1. Questo meccanismo di scambio dei dati è di estrema importanza ed influenza direttamente le prestazioni dell'intero *Annotation Server*. Una descrizione generale del sistema JDBC è fornita nel capitolo 6.

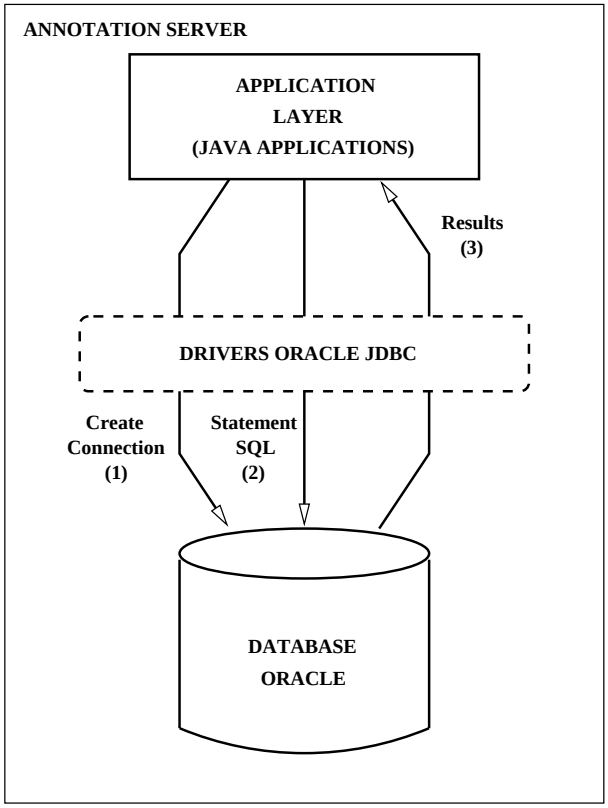

Figura 8.2: Operazioni del meccanismo JDBC.

Si possono identificare sostanzialmente tre operazioni fondamentali che possono essere eseguite utilizzando il meccanismo JDBC e che sono state realizzate nel prototipo proposto:

- Stabilire una connessione con il *Database*
- Spedire istruzioni SQL

[2]Tale interazione avviene a livello di *Annotation Server* come specificato nell'architettura descritta in modo dettagliato nella sezione 7.1.

- Processare i risultati

La figura 8.2 mostra schematicamente le operazioni di interazione tra lo strato applicativo e la base di dati, che sono i due componenti architetturali dell'*Annotation Server* (sezione 7.1). Il prototipo utilizza *drivers* proprietari per l'accesso alla base di dati *Oracle*.

Il primo passo consiste nella creazione di una connessione con la base di dati, ovvero di una sorta di canale di comunicazione tra lo strato applicativo e la base di dati sottostante (*step* (1)).

Successivamente le applicazioni Java possono inviare, attraverso la connessione precedentemente creata ed utilizzando i *drivers JDBC*, istruzioni SQL che verranno eseguite dal *DBMS Oracle* (*step* (2)).

Eseguite le operazioni necessarie a soddisfare le richieste ricevute il *DBMS Oracle* restituisce, sempre facendo uso della connessione esistente, i risultati allo strato applicativo (*step* (3)).

Il codice riportato nella figura 8.3 mostra un breve esempio in cui sono realizzati i tre passi mostrati nella figura 8.2 utilizzando il linguaggio Java.

```
Connection con = DriverManager.getConnection ("jdbc:protocol:
   username/password@host:port:subprotocol");
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT Ann_ID FROM Annota
                                      tion");
    while (rs.next()) {
      String s = getString("Ann_ID");
      }
```

Figura 8.3: Creazione di una connessione con la base di dati.

Il codice verrà spiegato in dettaglio nel seguito di tale sezione. A titolo di esempio nell'appendice C è riportata anche un'applicazione che si occupa della gestione della base di dati e che comprende quindi le istruzioni che realizzano il meccanismo di comunicazione attraverso JDBC.

La classe *DriverManager*

La classe *DriverManager* è lo strato di gestione di JDBC (fare riferimento al capitolo 6) e mantiene una lista delle classi *Driver* che sono registrate chiamando il metodo *DriverManager.registerDriver*. In un'applicazione che utilizza JDBC il primo passo da fare è quindi quello di registrare il driver necessario per accedere al *Database*. Nel caso del prototipo presentato in tale testo tale istruzione è la seguente:

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

Il metodo *Class.forName("nome del driver")* carica esplicitamente la classe del driver necessario alla comunicazione con il *Database*. Una volta che si è caricato il driver opportuno è possibile creare la connessione con il *Database* ed iniziare quindi la comunicazione.

Creazione di una connessione

Un oggetto *Connection* rappresenta una connessione con il *Database*. Una sessione di connessione include istruzioni SQL che vengono eseguite e risultati che vengono restituiti attraverso la connessione stessa. Una singola applicazione può avere una o più connessioni con un singolo *Database* o può avere connessioni con molti *Databases* diversi.

La modalità standard per stabilire una connessione con il *Database* è quella di chiamare il metodo *DriverManager.getConnection*. Questo metodo riceve come parametro d'ingresso l'URL del *Database*. La classe *DriverManager* mantiene una lista dei drivers registrati. Il seguente codice mostra come viene realizzata una connessione sull' *Annotation Server*:

```
String url ="jdbc:protocol:username/userpasswd@host:
port:subprotocol";
Connection con = DriverManager.getConnection(url);
```

L'URL (*Uniform Resource Locator*) fornisce tutte le informazioni necessarie per localizzare il *Database*. Un JDBC URL è sempre espresso nella seguente forma: *jdbc:subprotocol:subname* dove *jdbc* è il protocollo di comunicazione, *subprotocol* è il nome del particolare driver utilizzato e *subname* è l'informazione relativa al nome del *Database*, la porta di comunicazione, *username* e *userpassword*.

Statement

Un oggetto *Statement* è utilizzato per inviare istruzioni SQL al *Database*. Attualmente JDBC offre la possibilità di utilizzare tre diversi tipi di oggetti *Statement* per l'esecuzione di query dopo aver creato una connessione:

- *Statement*
- *PreparedStatement* che è una sottoclasse di *Statement*
- *CallableStatement* che è una sottoclasse di *PreparedStatement*

Un oggetto *Statement* è utilizzato per eseguire delle semplici *query* senza parametri; un oggetto *PreparedStatement* è utilizzato per eseguire istruzioni SQL precompilate con parametri in ingresso; tali istruzioni devono sempre avere parametri in ingresso altrimenti vengono eseguite facendo uso dell'oggetto *Statement*.

Un oggetto *CallableStatement* è utilizzato invece per eseguire chiamate a *Stored Procedure*.

ResultSet

Un oggetto della classe *ResultSet* contiene tutte le righe, di una determinata tabella che soddisfano una determinata *query* SQL; tale classe fornisce un insieme di metodi *getxxx* (dove xxx sta ad indicare il tipo dei dati contenuti in una stessa colonna) che consentono l'accesso alle varie colonne della riga corrente. Il metodo *ResultSet.next()* è utilizzato per muoversi alla riga successiva facendo in modo che la prossima riga diventi quella corrente.

Nella figura 8.4 è riportato un esempio di utilizzo della classe *ResultSet* e dei suoi metodi *next()* e *getxxx* necessari per accedere alle singole colonne di ciascuna riga di una tabella:

```
//select all the Projects of a Person
pstmt=conn.prepareStatement("SELECT DISTINCT idProject
FROM PERSONOFPROJECT WHERE idPerson=?");
// rs.getLong(2) is an identifier of a Person in the
   Database
pstmt.setLong(1,rs.getLong(2));
ResultSet rs2 = pstmt.executeQuery();
while (rs2.next()){
  value = new Long(rs2.getLong(1));
  projects.addElement(value);
}//end while
```

Figura 8.4: Esempio di *query* alla base di dati.

Come si può notare dall'esempio riportato nella figura 8.4 l'oggetto *ResultSet* mantiene un cursore che punta alla riga corrente della tabella specificata. Inizialmente tale cursore è posizionato prima della prima riga in modo tale che la prima chiamata del metodo *ResultSet.next()* determina lo spostamento del cursore sulla prima linea. Il cursore rimane valido sino a che l'oggetto *ResultSet* o l'oggetto *Statement* che l'ha generato non vengono cancellati. I metodi *getxxx* forniscono il valore della colonna specificata appartenente alla riga corrente. Vi sono tanti metodi *getxxx* quanti sono i tipi possibili che possono assumere i valori in una colonna.

8.1.3 RMI-*Remote Method Invocation*

Il meccanismo RMI (*Remote Method invocation*) è stato utilizzato nel prototipo quale meccanismo di comunicazione tra il *Proxy Server* e l'*Annotation Server* attraverso lo sviluppo di interfacce implementate da oggetti remoti che risiedono sul *Server* ed i cui metodi vengono invocati dai *Clients*. La descrizione generale del meccanismo RMI è riportata nel capitolo 6 relativo alle tecnologie utilizzate. In tale capitolo verranno descritti alcuni degli aspetti particolari relativi alla parte di codice utilizzata per realizzare i metodi remoti e renderli disponibili all'invocazione da parte dei *Clients*.

Regole alla base del meccanismo RMI

Per la realizzazione del meccanismo di comunicazione RMI devono essere effettuate tre operazioni principali:

- Scrivere i sorgenti Java
- Compilare e sviluppare i *files* che contengono le classi

- Attivare il *Remote Object Registry* ed il *Server*

In aggiunta a tali operazioni occorre definire ovviamente in modo corretto l'ambiente di sviluppo inizializzando opportunamente le variabili d'ambiente specifiche richieste dalla macchina virtuale Java.

La definizione dell'interfaccia è di estrema importanza in quanto in tal modo si stabiliscono le regole di comunicazione tra *Clients* che invocano i metodi remoti e *Server* su cui risiedono i metodi remoti.

Il prototipo poi prevede l'implementazione di tutti i metodi remoti definiti attraverso le interfacce; questo sostanzialmente è il cuore del lavoro svolto e la combinazione di tali metodi costituisce lo strato applicativo che risiede sull'*Annotation Server* e che gestisce il *Database* sottostante.

Nella figura 8.5 è riportata, a titolo di esempio, la parte di codice necessaria affinchè il *Client* possa identificare l'oggetto remoto *AnnotationServer* a cui appartengono tutti i principali metodi di gestione dell'*Annotation Server*.

Purtroppo non è possibile e non è nemmeno questa la sede per spiegare nei minimi dettagli il codice sopra riportato; l'aspetto che merita però particolare attenzione è il metodo *Naming.rebind("//hostname/objectname",obj)* con il quale il *Server* crea un riferimento per l'oggetto remoto *obj* che verrà poi utilizzato dal *Client* attraverso il metodo:

```
Naming.lookup(''//hostname/objectname'')}.
```

```
public static void main(String args[]){
// create and install a security manager
System.setSecurityManager(new RMISecurityManager());
try{
AnnotationServerImpl obj = new AnnotationServer
                     Impl("AnnotationServer");
Naming.rebind("//www.sts.tu-harburg.de/Annotation
       Server",obj);
System.out.println("AnnotationServer bound in registry");
```

Figura 8.5: Creazione di un riferimento per l'oggetto remoto.

Attivazione dell'*Annotation Server*

Dopo aver definito le interfacce e scritto i sorgenti che implementano i metodi remoti occorre compilare il codice prodotto. A questo punto è possibile creare i files *Stubs* e *Skeletons* che sono quelli che effettivamente consentono la comunicazione. La creazione di tali *files* avviene attraverso il compilatore *rmic* che fornisce *files bytecode*; lo *Stub* viene quindi portato sul *Client* (in caso contrario il *file Stub* viene caricato automaticamente la prima volta che il *Client* richiama l'oggetto remoto) mentre il *file Skeleton* resta sul *Server*.

Effettuate tali operazioni è possibile attivare il *Server* in modo tale che renda disponibili i metodi di oggetti remoti ai *Clients* che li vogliono richiamare. Ciò viene effettuato digitando il comando *rmiregistry* (questo comando si riferisce ad un sistema Solaris. Nel caso si utilizzi Windows 95 o Windows NT si dovrà digitare il comando *start rmiregistry*). Per default il registro viene attivato sulla porta 1099, ma è possibile anche modificare tale proprietà. A questo punto l'ultima operazione da svolgere è quella di attivare il *Server* con il comando:

```
java -Djava.rmi.server.codebase=http://hostname/directoryname/
nome_della_classe_che_implementa_i_metodi_remoti
```

Svolte tali operazioni il *Server* è a completa disposizione dei *Clients* che possono così richiamare i metodi remoti.

8.2 Esempi di implementazione di regole attive

Questa sezione propone alcuni esempi circa le modalità in base alle quali potrebbero essere implementate, facendo uso di *Trigger Oracle*, le regole attive classificate nella sezione 5.9.

Al fine della comprensione degli esempi riportati di seguito occorre fare riferimento allo schema logico della base di dati definito nella figura 5.20.

8.2.1 *Trigger Oracle* in una base di dati relazionale

La definizione di *trigger* fa parte del *Data Definition Language (DDL)* del linguaggio SQL; essi sono strumenti in base ai quali è possibile implementare regole attive e quindi sono basati sostanzialmente sul paradigma ECA (*Event-Condition-Action*) [S.Ceri *et al.* 1996].

In tale logica gli eventi che attivano il meccanismo dei *Trigger* sono le primitive per la manipolazione dei dati nel linguaggio SQL (*insert, delete, update*). La condizione è un predicato booleano espresso in SQL e l'azione è costituita da generiche primitive SQL che a loro volta potrebbero essere eventi di altre regole attive.

Nella definizione di *trigger* occorre prestare particolare attenzione a due aspetti che ne caratterizzano le modalità di funzionamento:

- Granularità
- Modalità di valutazione

Granularità

La granularità fa riferimento alla logica in base alla quale i *trigger* vengono attivati. Essi possono essere *row-level* oppure *statement-level*.

I *trigger* definiti *row-level* vengono attivati, a seguito del verificarsi dell'evento scatenante, su ogni tupla di una specifica tabella del modello relazionale e quindi, nel caso in cui la condizione sia verificata, viene eseguita l'azione prevista.

I *trigger* definiti *statement-level* vengono attivati sull'insieme di tuple che rappresentano il risultato della primitiva SQL che costituisce l'evento scatenante il *trigger* stesso.

Modalità di valutazione

La modalità di valutazione dei *trigger* può essere *immediata* oppure *differita*.

I *trigger* in modalità immediata vengono considerati subito dopo l'evento che li attiva. Questa è in genere la modalità standard di utilizzo dei *trigger*.

I *trigger* in modalità differita vengono invece valutati al momento del *commit* della transazione nell'ambito della quale si sono verificati i loro eventi scatenanti.

Nel caso specifico del sistema *Oracle* la modalità di valutazione dei *trigger* è immediata; inoltre il sistema *Oracle* garantisce un *rollback parziale* della primitiva scatenante e di tutte le azioni causate dal *trigger*. Ciò significa che se durante l'esecuzione del *trigger* viene sollevata qualche eccezione il sistema annulla tutti gli effetti che si sono verificati da quando il *trigger* è stato attivato.

Questi sono alcuni concetti generali relativi all'implementazione di regole attive mediante *trigger*; nel seguito di tale sezione sono riportati alcuni esempi di *Trigger Oracle* che implementano le regole attive classificate nella sezione 5.9.

Al fine della comprensione della sintassi utilizzata si consiglia di far riferimento a [S.Ceri *et al.* 1996], [Corporation 1992a], [Corporation 1992b].

8.2.2 Implementazione delle regole attive

La prima regola attiva di cui si riporta l'implementazione mediante *trigger Oracle* è quella relativa alla creazione di una nuova annotazione.

Il *trigger* riportato in figura 8.6 consente la determinazione automatica dell'insieme di indirizzi di posta elettronica degli utenti che devono essere informati circa la creazione di una nuova annotazione *Public*.

Questi utenti sono tutti coloro che sono interessati in modalità *Public* (SUBJECTOFPERSON.mod=0) ad almeno uno dei soggetti a cui fa riferimento la nuova annotazione.

In tal caso non è necessario controllare l'appartenenza degli utenti al progetto *Public* nell'ambito del quale è rilevante la nuova annotazione (new.idRelevantProject=0); questo perchè tutti gli utenti registrati fanno automaticamente parte del *Workspace Public*.

Ovviamente poi si dovranno implementare altre regole attive, in base alla stessa logica di quella riportata in figura 8.6, al fine di determinare gli utenti oggetto di notifica nel caso in cui l'annotazione sia rilevante nell'ambito di un determinato progetto diverso dal *Workspace Public*.

Altra situazione in cui è richiesta la notifica agli utenti tramite spedizione di *emails* è quella relativa alla cancellazione di un'annotazione.

```
create trigger Notifica
after insert on ANNOTATION
when (new.typology=0) AND (new.idRelevantProject=0)
for each row
  select distinct email
  from PERSON,SUBJECTOFPERSON
  where (SUBJECTOFPERSON.idPerson=PERSON.id) AND
        (SUBJECTOFPERSON.mod=0) AND
        SUBJECTOFPERSON.subjectName IN
              (SELECT subjectName
               FROM SUBJECTOFANNOTATION
               WHERE idAnnotation=new.id)
```

Figura 8.6: Regola attiva: creazione di un'annotazione *Public*.

In tal caso il proprietario dell'annotazione deve ricevere notizia circa il fatto che la sua annotazione è stata eliminata dal *System Administrator*. La regola attiva di figura 8.7 realizza tale funzionalità.

```
create trigger NotCancella
after delete on ANNOTATION
for each row
  select distinct email
  from PERSON
  where PERSON.id=old.idPerson
```

Figura 8.7: Regola attiva: cancellazione di un'annotazione.

Il *trigger* di figura 8.7 consente la determinazione dell'indirizzo di posta elettronica dell'utente che era proprietario dell'annotazione cancellata.

La regola di figura 8.7 deve essere considerata in combinazione al *trigger* la cui implementazione è riportata in figura 8.8.

```
create trigger CancellaGerarchia
after delete on ANNOTATION
for each row
  delete from ANNOTATION
  where (ANNOTATION.parent=old.id)
```

Figura 8.8: Regola attiva: cancellazione di una gerarchia di annotazioni.

Il *trigger* di figura 8.8 realizza la cancellazione in cascata delle annotazioni che si riferiscono all'annotazione originariamente eliminata dal *System Administrator*. La sezione 5.8.3 riporta la descrizione dettagliata del meccanismo ricorsivo di cancellazione delle annotazioni.

Questi brevi esempi hanno lo scopo di mostrare come sono implementate a livello di base di dati le regole attive classificate nella sezione 5.9 e che sono parte integrante del prototipo proposto.

In tal modo si realizza il principio di indipendenza della conoscenza, inglobando molte funzioni a livello di sistema per la gestione della base di dati e semplificando notevolmente l'implementazione dello strato applicativo soprastante che risulta essere costituito da tutte quelle funzionalità che non si possono configurare naturalmente come regole attive e descritte in linea di principio nel capitolo 7.

8.3 *Testing* ed ottimizzazione del codice

La fase di *Testing* delle applicazioni che costituiscono l'*Annotation Server* è organizzata in base ad una logica incrociata al fine di poter individuare la maggior parte degli errori effettuati durante l'implementazione.

In primo luogo si effettuano delle prove a livello locale simulando i metodi remoti e controllando la corretta interazione di questi ultimi con la base di dati e con le regole attive implementate con *Trigger Oracle*.

Il secondo passo è quello di attivare un *Server*, sul quale sono installati gli oggetti remoti, e di costruire delle applicazioni di *Testing* su di un *Browser* fittizio dotato di una rudimentale interfaccia grafica. Tali applicazioni sono realizzate al fine di richiamare e mandare in esecuzione, prima singolarmente e poi in combinazione, tutti i metodi remoti implementati.

Come riportato nella sezione 6.1, specificatamente nella parte relativa al processo seguito per lo sviluppo del software, il *Testing* non è effettuato al termine dell'implementazione bensì in concomitanza ad essa ovvero ogni volta che un nuovo metodo viene implementato. Ciò determina un notevole impiego di tempo ma consente l'ottenimento di migliori risultati.

La fase di *Testing* è seguita da una fase di ottimizzazione del codice prodotto. L'ottimizzazione dell'*Annotation Server* riguarda prevalentemente le modalità di accesso alla base di dati che maggiormente limitano le prestazioni dell'intero sistema.

La logica è quella di pensare delle *Query* che permettono di ottenere i dati voluti con il minor numero di accessi alla base di dati sottostante[3].

Altro aspetto a cui si deve prestare particolare attenzione è quello relativo all'implementazione dei metodi remoti richiamati dal *Proxy Server*; anche in tal caso la presenza di un numero eccessivo di metodi remoti determina il crollo delle prestazioni dell'intero sistema in quanto le connessioni di rete sono lente rispetto al caso di metodi eseguiti localmente. Per tal ragione devono essere eliminate eventuali ridondanze presenti nella prima versione implementata, senza però sacrificare alcuna delle funzionalità definite in fase di modellizzazione.

[3]Durante la fase di *Testing* si osserva infatti che l'aspetto più critico, dal punto di vista delle *performance*, è proprio quello relativo all'accesso alla base di dati.

8.4 Configurazione dell'*Annotation Server*

La procedura di configurazione dell'*Annotation Server* deve essere realizzata nel momento in cui si installa il sistema per la prima volta oppure nel momento in cui avviene qualche *crash* che ne blocca il funzionamento.

Oggetto	Tabella	Caratteristiche
Subject Public	**SUBJECT**	**Soggetto generico di nome "Public".**
System Administrator	**PERSON**	**Amministratore del sistema. Il nome di tale oggetto e' "administrator", la password e' "administrator" ed inoltre viene definito un email. Ovviamente tutti tali attributi possono essere modificati utilizzando gli opportuni metodi remoti disponibili.**
Workspace Public	**PROJECT**	**Generico spazio di lavoro radice dell'albero gerarchico dei progetti. L'amministratore del sistema e' anche amministratore del Workspace Public; il Workspace Public ha come parent il Workspace Public medesimo.**
Link System Administ. => Workspace Public	**PERSONOFPROJECT**	**Legame tra amministratore del sistema e Workspace Public di cui ne e' membro nonche' amministratore.**
Link Subject Public => Workspace Public	**SUBJECTOFPROJECT**	**Ogni oggetto della classe PROJECT deve far riferimento ad un oggetto della classe SUBJECT. Il Workspace Public ha come soggetto di riferimento il SubjectPublic**
Document Public	**DOCUMENT**	**Documento virtuale identificato per mezzo di un URL simbolicamente chiamato Public.**
Link Subject Public => System Administ.	**SUBJECTOFPERSON**	**Legame tra System Administrator e soggetto Public. Ogni entita' della classe PERSON deve fare riferimento ad un soggetto. Il System Administrator fa riferimento al generico soggetto Public in modalita' Public.**
Annotation Public	**ANNOTATION**	**Generica annotazione il cui titolo e' Public. Tale annotazione e' di tipo Public, fa riferimento al documento Public, ha come author il System Administrator,appartiene al Workspace Public ed e' rilevante per il Workspace Public.**
Link Subject Public => Annotation Public	**SUBJECTOFANNOTATION**	**Ogni annotazione deve avere almeno un soggetto a cui si riferisce. L'annotazione Public fa riferimento al generico soggetto Public.**

Tabella 8.1: Oggetti creati in fase di inizializzazione dell'*Annotation Server*.

La prima operazione da svolgere consiste nella creazione della base di dati, ovvero delle tabelle che realizzano gli oggetti definiti in fase di modellizazione. Ciò avviene eseguendo un'applicazione java (*CreateTables.java*) che automaticamente, utilizzando il meccanismo JDBC ed il DDL (*Data Definition Language*) di SQL, crea tutte le tabelle necessarie.

Fatto ciò occorre eseguire un'altra applicazione java (*InsertValues.java*) la quale inizializza i valori delle tabelle con delle tuple sulla base delle quali il sistema potrà cominciare a funzionare. Con tale applicazione viene creato il *Workspace Public* ed il *System Administrator* che è definito automaticamente membro ed amministratore del progetto *Public*. Viene inoltre creato un soggetto fittizio identificato dal nome *Public* a cui fanno riferimento il *Workspace Public* ed il *System Administrator*.

La tabella 8.1 riporta schematicamente i vari oggetti, istanze delle classi implementate, che vengono creati automaticamente nel momento in cui viene eseguita la procedura di *Start*[4] dell' *Annotation Server*.

Ultima operazione da svolgere in fase di inizializzazione dell'*Annotation Server* è l'esecuzione dell'applicazione *GarbageCollector.java* la cui funzionalità è spiegata dettagliatamente nella sezione 7.2.

Tutti i parametri di configurazione del sistema sono specificati nel file *AnnotationSystemConfiguration.txt*; questo file contiene tutte le informazioni relative alla configurazione dell'intero sistema da un punto di vista tecnico. In tale modo è possibile definire il nome del *driver* da utilizzare per accedere alla base di dati, l'indirizzo dell'*Annotation Server*, i nomi con cui referenziare gli oggetti remoti, il nome e la porta del *Server* da utilizzare per spedire le *emails* ed i tempi in base ai quali realizzare la procedura di *Garbage Collection* della tabella *KEYS*.

Le applicazioni, che sono sempre attive sull'*Annotation Server* sotto forma di metodi remoti, utilizzano i parametri specificati nel file *AnnotationSystemConfiguration.txt* realizzando le funzionalità a lungo discusse nel capitolo 5.

Tale meccanismo di configurazione è stato pensato al fine di garantire una maggior flessibilità dell'intero sistema consentendo una semplice e soprattutto rapida modifica delle sue caratteristiche tecniche.

8.5 Domini applicativi del prototipo

Questa sezione descrive brevemente quali sono i campi in cui può trovare concreta applicazione il sistema per l'annotazione di documenti multimediali descritto in tale testo. Particolare attenzione sarà dedicata al progetto *WEL* (sezione 4.2.2) nel contesto del quale si inserisce e trova la sua naturale applicazione il lavoro eseguito.

8.5.1 Approcci alla personalizzazione e cooperazione

Il sistema di annotazione sviluppato costituisce uno strumento essenziale per la realizzazione di applicazioni personalizzate e per lo sviluppo di un lavoro cooperativo di gruppo.

[4]Tale procedura viene eseguita dal responsabile del sistema al momento dell'installazione od a seguito del verificarsi di guasti. I passi da seguire sono riportati nel file *StartServer.txt*.

Attraverso il meccanismo delle *Private Annotations* ogni utente può aggiungere commenti e considerazioni personali a ciascun documento presente sul WWW senza che nessun altro li possa consultare ed utilizzare. Ciò è un aspetto di estrema importanza ed utilità, che consente l'elaborazione di documenti personalizzati attraverso l'aggiunta di meta-informazioni quali sono le annotazioni.

In aggiunta a ciò il sistema offre la possibilità di effettuare delle *Public Annotations* e delle *Project Annotations*. Con tali meccanismi ogni persona registrata nel sistema, ed appartenente a determinati progetti di lavoro, può formulare delle considerazioni che mette a disposizione di un particolare gruppo di progetto. La logica in base alla quale è possibile effettuare le *Project Annotations* è basata essenzialmente sulla gerarchia dei progetti esistenti.

Ogni *Project Annotation* ha un progetto a cui appartiene, che è quello nell'ambito del quale l'annotazione è stata creata, ed un progetto per cui è rilevante il quale viene specificato dal suo autore. Il progetto per cui un'annotazione è rilevante può essere solamente uno dei progetti che si trovano percorrendo la gerarchia dal basso verso l'alto partendo dal progetto di appartenenza. Tutte le persone che appartengono al progetto per cui un'annotazione è rilevante possono leggere ed annotare tale annotazione.

Questa logica di funzionamento contribuisce alla realizzazione di un lavoro cooperativo; attraverso la formulazione di *Project Annotations* e l'annotazione di annotazioni si possono creare dei dialoghi multiutente che favoriscono lo scambio di informazioni.

Tutto ciò viene integrato con meccanismi di notificazione tramite *e-mails*, attraverso i quali gli utenti vengono messi al corrente di tutte le modifiche (aggiunte o cancellazioni) effettuate sugli insiemi di annotazioni a cui sono interessati.

8.5.2 Progetto WEL-*Warburg Electronic Library*

L'ambito nel quale verrà utilizzato il sistema realizzato è il progetto WEL che prevede la realizzazione di una biblioteca digitale per la libreria "Warburg-Haus", avente una sede ad Hamburg ed una sede a Londra.

Il progetto *Warburg Electronic Library* (WEL) è iniziato nell'ambito di un lavoro di collaborazione interdisciplinare [Niederée *et al.* 1996] tra il gruppo di ricerca della *Technische Universität Hamburg-Harburg* e l'*Art History Department* dell'universita' di Hamburg.

L'obiettivo principale di tale progetto consiste nell'esame, sviluppo ed applicazione delle librerie digitali per scopi di ricerca storica.

Il dominio applicativo della storia dell'arte è caratterizzato da immagini arricchite con del testo e da altri oggetti multimediali quali filmati oppure documenti audio. La ricerca nel campo della storia dell'arte si focalizza sull'esame di tali immagini, sull'identificazione di icone rappresentative eventi e/o persone e sulla loro classificazione in base a logiche ben definite.

Il progetto WEL presenta diversi aspetti tra cui i due principali riguardano da un lato la realizzazione di un indice per la ricerca iconografica su temi politici e dall'altro lo sviluppo di un sistema per l'annotazione di documenti multimediali in spazi condivisi.

La creazione di un indice iconografico su tematiche riguardanti la politica (*Political Iconography - Index*) è competenza di quell'area della Storia dell'Arte che si occupa di esaminare e

comprendere i messaggi politici contenuti in immagini raffiguranti regnanti, politici, cerimonie etc.

L'assunzione principale alla base di tale processo consiste nel fatto che gli effetti che conseguono atti politici non sono solo contenuti in documenti scritti ma anche in immagini di dipinti, monumenti e sculture. Ciò giustifica la realizzazione di un indice attraverso il quale classificare e gestire questo complesso materiale iconografico.

In tale contesto un sistema a supporto dell'annotazione di documenti multimediali risulta essere molto importante e di grande utilità pratica. Attraverso l'annotazione sarà possibile aggiungere commenti e considerazioni a documenti, per di più di carattere storico-politico, memorizzati nella base di dati della biblioteca e messi a disposizione attraverso il *Server WWW*.

È semplice capire come, in tale contesto, la possibilità di annotare documenti multimediali consente un'elaborazione attiva dei contenuti, generalmente necessaria nel momento in cui si effettua una lettura a fini di studio e non solo a carattere ricreativo[5].

Nonostante tale ambito applicativo specifico il sistema di annotazione realizzato può avere numerose altre applicazioni a carattere generale; l'organizzazione degli utenti in gruppi di progetto strutturati gerarchicamente e la definizione dei diritti di accesso per le annotazioni offrono la possibilità di utilizzare tale sistema anche in ambito universitario oppure industriale al fine di ottenere cooperazione e scambio di informazioni tra i componenti dei vari gruppi.

8.5.3 Potenzialità del sistema

Descrivere le potenzialità del sistema è sempre molto difficile, soprattutto se lo si è sviluppato in prima persona, in quanto si può rischiare di enfatizzare eccessivamente certi aspetti a scapito di altri.

Personalmente ritengo che gli elementi che più possono essere messi in evidenza sono i meccanismi di notificazione implementati, la possibilità offerta agli utenti di realizzare annotazioni private o di progetto, l'organizzazione gerarchica dei gruppi di utenti, i meccanismi di annotazione ricorsiva e le modalità di ricerca delle annotazioni che risultano essere particolarmente flessibili.

In aggiunta a tutto ciò occorre fare una breve citazione alle due tecniche, RMI (*Remote Method Invocation*) e JDBC, attraverso cui è stata realizzata la comunicazione tra *Proxy-Server* ed *Annotation Server* e l'interazione tra l'*Annotation Server* e la base di dati. In fase di ottimizzazione del codice si è prestata molta attenzione al fatto di riscrivere i metodi realizzati in modo tale che sfruttassero nel miglior modo possibile tali tecniche di comunicazione ottenendo in tal modo delle prestazioni complessive notevolmente migliori.

8.5.4 Restrizioni e limiti

Ovviamente, come in ogni progetto, occorre anche mettere in evidenza quegli aspetti che sono stati trascurati od almeno non trattati in maniera esauriente e che quindi costituiscono oggetto di

[5]Per avere informazioni più approfondite per quanto riguarda il progetto WEL è possibile fare riferimento ai seguenti indirizzi: *http://www.sts.tu-harburg.de/projects/WEL/entry.html* ed anche *http://www.warburg-haus.-hamburg.de/* ed alla sezione 4.2.2.

ulteriore sviluppo e studio per la realizzazione di successive versioni.

Un primo elemento riguarda il meccanismo di sicurezza realizzato; la scelta effettuata è stata quella di realizzare il meccanismo classico attraverso cui l'utente viene identificato per mezzo del binomio *userName + userPassword*. Tale modalità di autentificazione potrebbe essere ripensata in modo tale da garantire la maggior sicurezza possibile, probabilmente anche attraverso la realizzazione di un *Firewall* che protegga l'intero sistema da indesiderate intrusioni esterne.

Altro aspetto che potrebbe essere ripensato è il meccanismo di notificazione a seguito di cancellazione di un'annotazione. La soluzione adottata è una delle possibili; purtroppo tale decisione è influenzata enormemente dall'ambito applicativo in cui il sistema viene utilizzato e dalla semantica delle annotazioni. Si potrebbe pensare ad esempio di realizzare diversi meccanismi di notificazione dando la possibilità al *System Administrator* di scegliere quello che ritiene più opportuno nel contesto specifico di utilizzazione del sistema.

Capitolo 9

Conclusioni

In tale capitolo è riportato un breve riassunto dei concetti principali trattati nel corso dell'intero testo; inoltre si fa accenno ai possibili sviluppi futuri che potrà avere il sistema progettato, per concludere con alcune osservazioni personali relative all'esperienza vissuta ad Hamburg durante il periodo di progettazione ed implementazione del prototipo.

9.1 Riepilogo

Il sistema per l'annotazione di documenti multimediali proposto in tale testo è il frutto di una lunga fase di analisi nel corso della quale si sono prese precise decisioni di fronte a situazioni che, inevitabilmente, implicano dei compromessi. La soluzione proposta alle varie questioni non è sicuramente l'unica possibile, ma si è cercato comunque di raggiungere il miglior risultato possibile[1].

Di centrale importanza è stata la progettazione dell'intero sistema e la definizione dell'architettura di supporto; l'implementazione, incentrata particolarmente sull'*Annotation Server* e sui meccanismi di comunicazione, ha occupato la porzione temporale più rilevante nell'ambito dell'intero lavoro.

Da sottolineare l'implementazione dei meccanismi di notificazione realizzati sull'*Annotation Server* che lo rendono un componente attivo nell'ambito dell'intero sistema, capace di reagire a determinati eventi con l'esecuzione di specifiche azioni a seguito della soddisfazione di determinate condizioni.

Estremamente importante è anche il meccanismo di chiamata di metodi remoti (*RMI-Remote Method Invocation*) attraverso il quale il *Proxy-Server* può usufruire dei servizi messi a disposizione dall'*Annotation Server*[2].

In fine occorre ricordare la presenza di una base di dati attraverso la quale vengono gestite

[1] In termini di logica di funzionamento e di prestazioni.

[2] Tutti i servizi messi a disposizione dall'*Annotation Server* sono realizzati come metodi di oggetti remoti. In tal modo è possibile avere anche più *Proxy Servers* che usufruiscono dei servizi remoti disponibili sull'*Annotation Server*.

e mantenute le annotazioni, ed anche la realizzazione di una interfaccia grafica (*Graphical User Interface*) grazie alla quale gli utenti possono interagire con il sistema in modo semplice e rapido.

La realizzazione dell'intero sistema è stata guidata dai principi fondamentali di *personalizzazione* e *cooperazione*, che sono concetti basilari per un meccanismo di annotazione a documenti multimediali. Attraverso la possibilità di realizzare delle annotazioni ogni utente può personalizzare un qualsiasi documento presente sul WWW, arricchendolo con commenti e considerazioni che nessuna altra persona potrà leggere ed utilizzare. D'altro canto il sistema proposto offre l'opportunità di creare delle *Project Annotations*, che costituiscono il presupposto per l'istaurarsi di un lavoro cooperativo e di gruppo nell'ambito di un determinato progetto o insieme di progetti. Questi sono concetti fondamentali che non devono essere persi di vista nello sviluppo di un sistema per l'annotazione.

9.2 Sviluppi futuri

Il prototipo proposto si inserisce nell'ambito del progetto WEL (*Warburg Electronic Library*, sezione 4.2.2) il quale prevede la realizzazione di una biblioteca digitale per la gestione di materiale storico costituito per la maggior parte da immagini. In un tale contesto è estremamente importante avere anche un meccanismo di annotazione dei documenti multimediali al fine di utilizzare in modo attivo il materiale raccolto.

Tale progetto avrà la sua conclusione nel 2002; ciò significa che anche il sistema realizzato a supporto dell'annotazione avrà delle versioni successive in cui verranno implementati altri aspetti che nel corso del tempo potranno emergere. Ciò nonostante il sistema di annotazione proposto può essere utilizzato anche in altre aree come descritto nel capitolo 8.

Una funzionalità aggiuntiva potrebbe essere la realizzazione di un meccanismo di *Garbage Collection* di tutte quelle annotazioni che risiedono da un tempo eccessivamente elevato nel sistema[3]. Nella versione attuale tale compito è lasciato alla discrezionalità del *System Administrator* il quale può decidere autonomamente quando e quali annotazioni cancellare; l'automatizzazione di tale meccanismo è sicuramente un aspetto interessante che in futuro potrà essere oggetto di ulteriori analisi.

Altro aspetto riguarda l'implementazione del meccanismo di notifica a seguito di cancellazione di un'annotazione; nel prototipo presentato si è deciso di cancellare tutta la gerarchia di annotazioni sottostante notificando il proprietario di ciascuna di esse con una copia dell'annotazione direttamente soprastante. In futuro si potrebbero implementare altre soluzioni come proposto nella sezione 5.8.3 in base alla logica che più si ritiene adeguata. Oppure ancora si potrebbe configurare l'*Annotation Server* in base a certe specifiche dando la possibilità al *System Administrator* di modificare i parametri di configurazione in base alle esigenze prevalenti degli utenti del sistema in un determinato periodo di tempo[4].

[3] Tali annotazioni possono essere cancellate in quanto prive di significato per il fatto che sono state realizzate da molto tempo e forse anche nell'ambito di progetti ormai conclusi.

[4] Ciò significa che il *System Administrator* avrebbe la possibilità di modificare l'algoritmo di notificazione in base ai criteri che più ritiene adeguati nell'ambito del sistema che ha la responsabilità di gestire.

Inoltre sarebbe interessante estendere le funzionalità del sistema aggiungendo la possibilità di effettuare diversi tipi di annotazioni come proposto nella sezione 3.5.2 del capitolo 3. Questo comporterebbe una modifica della modalità implementativa del *Proxy Server* e dell'interfaccia utente lasciando però inalterati i principi sulla base dei quali è stato progettato l'*Annotation Server*.

In aggiunta a queste possibili evoluzioni future definite in modo specifico si possono identificare alcuni ambiti di sviluppo a carattere generale che forniscono concetti in base ai quali poter arricchire il sistema per l'annotazione descritto in tale testo:

Basi di dati attive: l'intero sistema potrebbe essere progettato facendo uso di un *Database* attivo che consente l'implementazione di regole attive sulla base delle indicazioni riportate nella sezione 5.9.

In tal caso lo strato applicativo realizzato sull'*Annotation Server* viene notevolmente semplificato e molte funzionalità sono inglobate nella progettazione ed implementazione della base di dati attiva.

Basi di dati distribuite: lo logica della distribuzione dei dati rappresenta sicuramente una delle tendenze più moderne in campo informatico.

La soluzione che prevede l'utilizzo di una base di dati distribuita può migliorare notevolmente le prestazioni del sistema di annotazione, estendendo notevolmente i suoi ambiti di applicabilità e soprattutto migliorando le suo proprietà di scalabilità.

Basi di dati ad oggetti: anche la tecnologia delle basi di dati ad oggetti rappresenta una nuova filosofia in base alla quale gestire i dati seguendo il paradigma ad oggetti degli attuali linguaggi di programmazione.

La base di dati relazionale, attualmente gestita dall'*Annotation Server* (sezione 7.1), potrebbe essere sostituita con una nuova base di dati ad oggetti, nel rispetto delle specifiche che si sono seguite in fase di implementazione del sistema e riportate nel capitolo 8.

Interfaccia Grafica Personalizzata: in funzione al particolare dominio applicativo del sistema di annotazione si può procedere alla realizzazione di un'interfaccia grafica personalizzata sulla base delle richieste e delle esigenze dello specifico utente.

Annotazioni come servizi: in futuro si potrebbe estendere il concetto di annotazione arricchendolo di connotazioni proprie del dominio informatico.

A tal proposito un'annotazione potrebbe essere considerata anche come un servizio associato ad un documento od a sue parti interne quali porzioni di testo od immagini. Un servizio non è altro che una procedura che svolge determinate operazioni; un esempio potrebbe essere il servizio di ricerca che è molto utile in correlazione ai documenti testuali al fine di ritrovare determinate parole o frasi in essi contenute. Con tale estensione il concetto di annotazione si arricchisce notevolmente aprendo nuovi e più ampi campi di ricerca.

9.3 Conclusione

Arrivato a questo punto vorrei esprimere il mio più sentito ringraziamento a tutto il personale del Dipartimento di Informatica della Technische Universität Hamburg-Harburg che, con grande pazienza, mi ha aiutato a superare le inevitabili barriere che vi sono quando si parla una lingua straniera.

L'esperienza che ho vissuto nei sette mesi trascorsi ad Hamburg è stata straordinaria; ci sono stati dei momenti molto difficili, soprattutto legati alla lontananza dalle persone a cui voglio un bene immenso. Nonostante ciò, dal punto di vista professionale, ho imparato moltissimo in questo periodo e la soddisfazione che ho provato nel momento in cui ho portato a conclusione il mio lavoro, presentando ad Hamburg nel corso di un seminario una DEMO del mio prototipo, è stata veramente indescrivibile.

Spero che i contenuti riportati in questo testo possano servire in futuro a chiunque desideri progettare un nuovo sistema per l'annotazione di documenti multimediali, affinché le soluzioni proposte nel mio progetto possano essere migliorate contribuendo in tal modo alla continua evoluzione della scienza dell'informazione.

Appendice A

Glossario

Annotation Author è il ruolo che identifica un generico utente del sistema che crea un'annotazione ad un documento od ad un'altra annotazione.

Annotation Reader è il ruolo di un generico utente del sistema che legge un'annotazione od una serie di annotazioni precedentemente ricercate in base ai criteri disponibili.

Annotation System è il sistema per mezzo del quale gli utenti possono effettuare delle annotazioni a documenti multimediali presenti ovunque sul World Wide Web.

Annotations Server *server* che si occupa della gestione del *Database* contenente le informazioni relative alle annotazioni, agli utenti ed ai progetti; elemento essenziale nell'architettura del sistema di annotazione proposto.

Annotazione rielaborazione personale e critica del contenuto di un documento multimediale presente sulla rete internet. Le annotazioni costituiscono un presupposto ed un elemento essenziale a supporto dei meccanismi di personalizzazione e di cooperazione.

Browser componente dell'architettura proposta attraverso il quale l'utente interagisce con il sistema; sul *browser* è realizzata l'interfaccia utente GUI (*Graphical User Interface*). Il *browser* comunica direttamente con il *Proxy Server* per reperire i documenti dalla rete e per richiamare i metodi remoti che risiedono sull'*Annotation Server*.

Cooperazione concetto a carattere generale che sta ad indicare una modalità in base alla quale più persone svolgono un determinato compito condividendo spazi di lavoro e domini di interesse al fine di aumentare l'efficienza e la qualità del prodotto ottenuto.

DBMS acronimo di *Data Base Management System*; è l'unità che si occupa direttamente della gestione del *Database*. Il meccanismo JDBC consente di costruire un dialogo tra le applicazioni Java ed il particolare DBMS adottato (ad esempio *Oracle DBMS*).

Interfaccia utente meccanismo per mezzo del quale è resa possibile l'interazione tra l'utente ed il sistema di elaborazione; generalmente l'interfaccia utente è grafica (GUI-*Graphical*

User Interface) ed *User Friendly* ovvero realizzata in modo tale da rendere il più semplice possibile l'utilizzo del sistema da parte dell'utente.

Java linguaggio di programmazione *Object-Oriented*; tale linguaggio introduce nuovi concetti arricchendo notevolmente le potenzialità del WWW ed aprendo nuove frontiere fornendo l'opportunità agli utenti dei *Servers WWW* di essere più interattivi e di giocare un ruolo attivo nell'utilizzo dei documenti disponibili.

JDBC meccanismo che definisce il protocollo di comunicazione tra applicazioni scritte in linguaggio Java ed un qualsiasi *Database* sottostante. Tale comunicazione è realizzata tramite l'utilizzo di appositi *drivers*. JDBC è un marchio di fabbrica registrato dalla Sun Microsystem ma molto spesso viene considerato acronimo di *Java Database Connectivity*.

JDBC OCI *drivers* che forniscono un'implementazione per le interfacce JDBC che utilizzano l'interfaccia OCI (*Oracle Call Interface*) per interagire con un *Database Oracle*. I *drivers* JDBC OCI sono stati realizzati per essere utilizzati esclusivamente da applicazioni scritte in linguaggio Java.

JDBC thin *driver* per la comunicazione con un *Database Oracle* ed utilizzato dal meccanismo JDBC per l'interazione con applicazioni ed Applet Java.

Notificazione è il meccanismo principale che caratterizza l'*Annotation Server*; la notificazione viene realizzata tramite spedizione di *e-mails* a seguito della creazione di una nuova annotazione o della cancellazione di una esistente.

Progetto nel prototipo proposto il termine progetto fa riferimento ad un qualsiasi gruppo di persone legate da interessi prevalentemente comuni e che lavorano per il conseguimento di determinati obiettivi. I progetti sono organizzati in una struttura ad albero in cui il nodo radice è il *Workspace Public*.

Project Administrator è l'utente che ha creato il progetto e che quindi ha i diritti per poterlo gestire ad esempio aggiungendo od eliminando membri al gruppo iniziale.

Personalizzazione principio che sta ad indicare la possibilità offerta agli utenti di un determinato sistema di utilizzare le risorse disponibili in modo personalizzato. In tale ottica ogni utente può costruirsi delle risorse "su misura" da utilizzare poi nello svolgimento del proprio compito. Il meccanismo di annotazione, ad esempio, offre la possibilità di personalizzare documenti multimediali tramite l'aggiunta di commenti e considerazioni.

Proxy Server è un componente dell'architettura del sistema proposto attraverso cui il *Browser* può comunicare con il mondo esterno e con l'*Annotation Server*. Nel *Proxy Server* avviene il *merge* tra le annotazioni ed il documento a cui si riferiscono.

RMI acronimo di *Remote Method Invocation*; definisce le modalità in base alle quali applicazioni Java possono comunicare con oggetti remoti invocando i loro metodi. In un modello *Client/Server* i metodi remoti risiedono sul *Server* e vengono invocati dai vari *Clients*.

SQL acronimo di *Structured Query Language*; linguaggio ormai standard per effettuare *query* e definire tabelle per basi di dati relazionali.

Subject è il termine utilizzato per definire un generico tema di studio; ogni progetto ed ogni annotazione devono avere almeno un *Subject* a cui si riferiscono. Anche gli utenti hanno dei *Subjects* a cui sono interessati e che possono gestire in maniera molto flessibile.

System Administrator è colui che amministra l'intero sistema di annotazione; solo il *System Administrator* ha la possibilità di cancellare le annotazioni dal sistema.

UML acronimo di *Unified Modeling Language*. L'UML è un linguaggio per la definizione di modelli e non un metodo; questo perché esso definisce le regole sintattiche (grafiche) in base alle quali costruire un modello senza però specificare il processo da seguire per raggiungere tale obiettivo. UML è un linguaggio ortogonale al processo di sviluppo.

URL acronimo di *Uniform Resource Locator* e rappresenta un meccanismo di identificazione univoca delle risorse presenti sulla rete. JDBC ed RMI utilizzano URLs per identificare rispettivamente il *Database* e gli oggetti remoti con cui dialogare.

Utente è un generico oggetto della classe PERSON che viene creato utilizzando la maschera di registrazione degli utenti al sistema. Per *default* un utente fa parte, sin dal momento in cui viene creato, del *Workspace Public*. Una volta registrato l'utente può poi entrare a far parte del gruppo di lavoro dei diversi progetti presenti nell'albero gerarchico mantenuto dal sistema.

World Wide Web struttura distribuita a livello mondiale basata sulla rete di comunicazione *Internet*; il WWW si basa su di un modello *Client/Server* attraverso cui più *Clients* possono chiedere dei documenti ai *Servers* WWW che li mettono a disposizione.

Appendice B

Il Modello

B.1 *Use cases*

Di seguito sono riportati gli *use cases* che costituiscono parte integrante del capitolo 5. Per ogni *use case* è riportata una breve descrizione per meglio comprenderne il significato.

In primo luogo è opportuno ricordare quali sono gli attori individuati in fase di modellizzazione del sistema:

Annotation Server: l'*Annotation Server* è un componente dell'intero sistema (per avere maggiori informazioni riguardanti l'architettura del sistema fare riferimento alla sezione 7.1). L'*Annotation Server* è un componente "reattivo"[1] in quanto realizza il meccanismo di notificazione oltre a tutte le attività di gestione della base di dati.

User: l'utente è una qualsiasi persona registrata nel sistema; più in dettaglio si possono individuare diversi ruoli che, all'interno del sistema, hanno dei diritti e quindi sono abilitati a svolgere compiti diversi.

Annotation Reader: colui che legge un determinato insieme di annotazioni.

Annotation Author: colui che crea una determinata annotazione riferita ad un documento od ad un'altra annotazione.

Project Administrator: tale attore rappresenta l'amministratore del progetto; il *Project Administrator* non coincide necessariamente con una figura fisica ma rappresenta sostanzialmente un ruolo attraverso cui vengono gestiti tutti gli aspetti relativi ad un determinato progetto.

System Administrator: anche questo è un ruolo sostanzialmente amministrativo; il *System Administrator* è l'amministratore del *Workspace Public* nonché amministratore dell'intero sistema.

[1] Ciò significa che l'*Annotation Server* è in grado di eseguire determinate operazioni a seguito del verificarsi di particolari eventi che rispettano specifiche condizioni.

Segue ora un elenco di tutti gli *Use Cases* che rappresentano l'insieme completo delle funzionalità offerte dal sistema.

- **Creazione di una nuova annotazione**

  La figura B.1 rappresenta la funzionalità di creazione di una nuova annotazione; da notare il meccanismo di identificazione dell'utente che realizza la nuova annotazione e che deve quindi essere stato registrato precedentemente nel sistema.

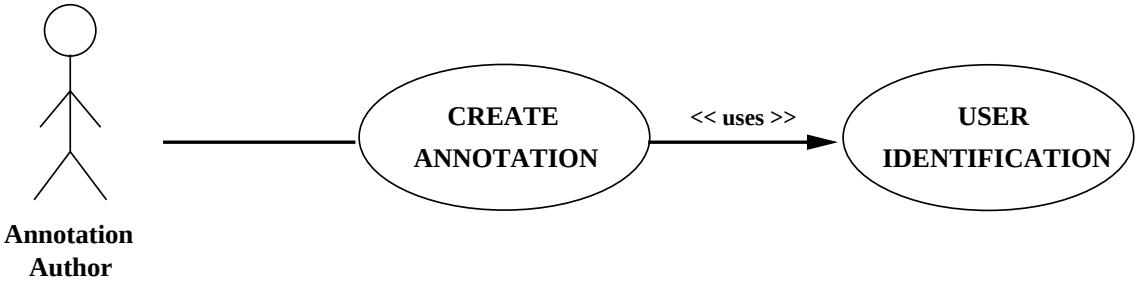

Figura B.1: Creazione di una nuova annotazione.

La figura B.2 mostra più in dettaglio i passi attraverso cui un utente effettua la creazione di una nuova annotazione.

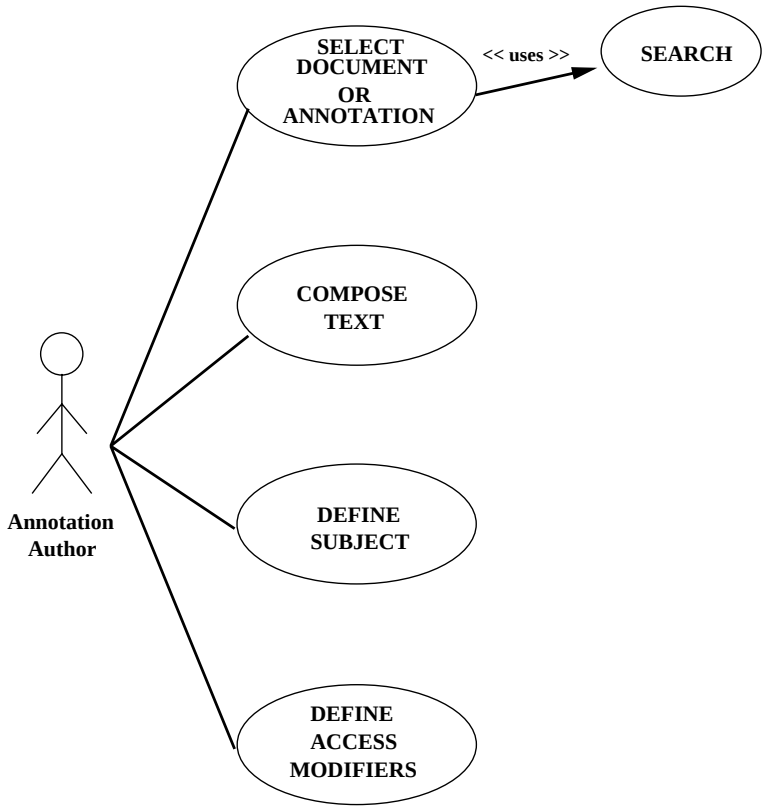

Figura B.2: Creazione di una nuova annotazione -Dettagli-.

Anzitutto viene selezionato il documento o l'annotazione che si desidera annotare; quindi, una volta trovato il documento desiderato, si crea l'annotazione definendone il testo, i soggetti a cui si riferisce e gli *Access Modifiers*. A questo punto l'*Annotation Server* attiva la procedura di notificazione in base alle regole spiegate nella sezione 5.8.2.

- **Creazione di una nuova annotazione riferita a parti del documento annotato**

  Questo *use case* rappresenta in dettaglio la funzionalità generale di creazione di un'annotazione riportata nella figura B.1 nel caso in cui la nuova annotazione sia riferita a parti interne dell'intero documento quali porzioni di testo e/o immagini. La figura B.3 presenta i vari passi compiuti dall'autore della nuova annotazione attraverso l'interazione con il sistema.

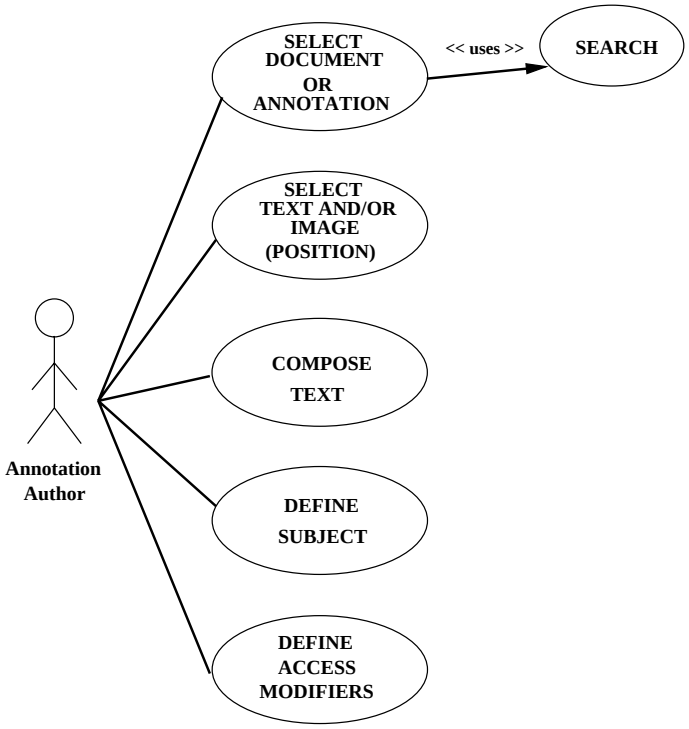

Figura B.3: Creazione di una nuova annotazione riferita a parti del documento-Dettagli-.

- **Visualizzazione di una serie di annotazioni**

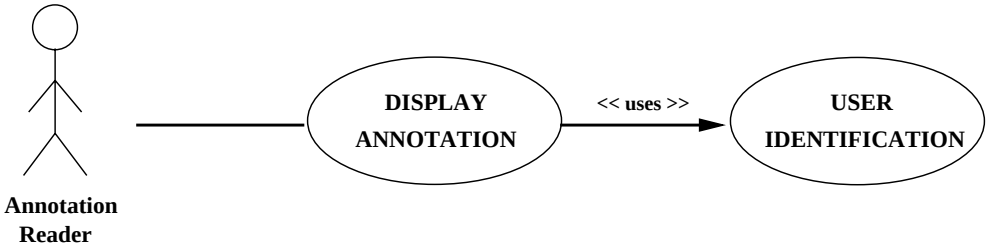

Figura B.4: Visualizzazione di una serie di annotazioni.

La figura B.4 mostra la funzionalità di visualizzazione delle annotazioni messa a disposizione dal sistema.

- **Visualizzazione delle annotazioni riferite ad un documento**

  Ogni utente registrato nel sistema deve avere la possibilità di visualizzare le annotazioni relative ad un determinato documento od ad una determinata annotazione (figura B.5). Ovviamente il sistema dovrà controllare l'identità dell'utente in modo tale da permettere la visualizzazione delle sole annotazioni su di cui l'utente possiede diritti di lettura.

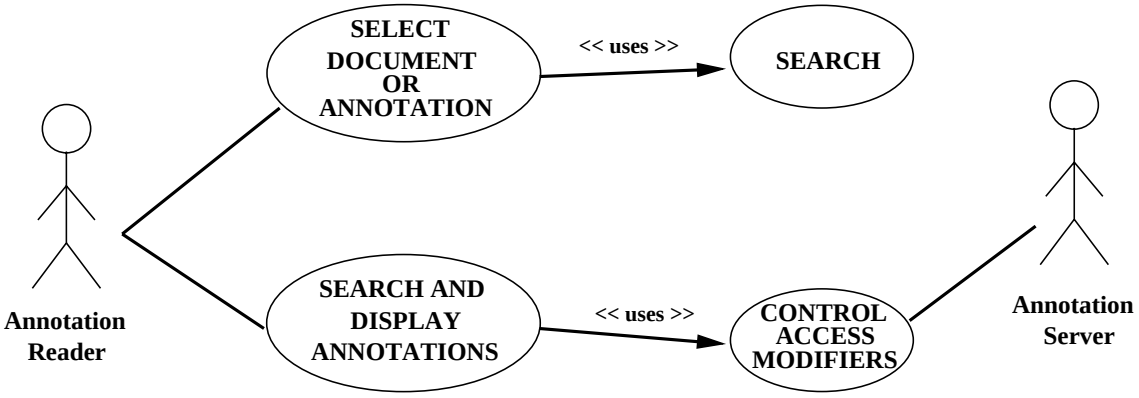

Figura B.5: Visualizzazione delle annotazioni riferite ad un documento.

- **Visualizzazione delle annotazioni con un soggetto specificato**

  In tale situazione (figura B.6) l'utente seleziona un determinato *Subject* ed il sistema ricerca tutte le annotazioni che hanno almeno quel soggetto compreso nell'insieme dei soggetti appartenente ad ogni annotazione.

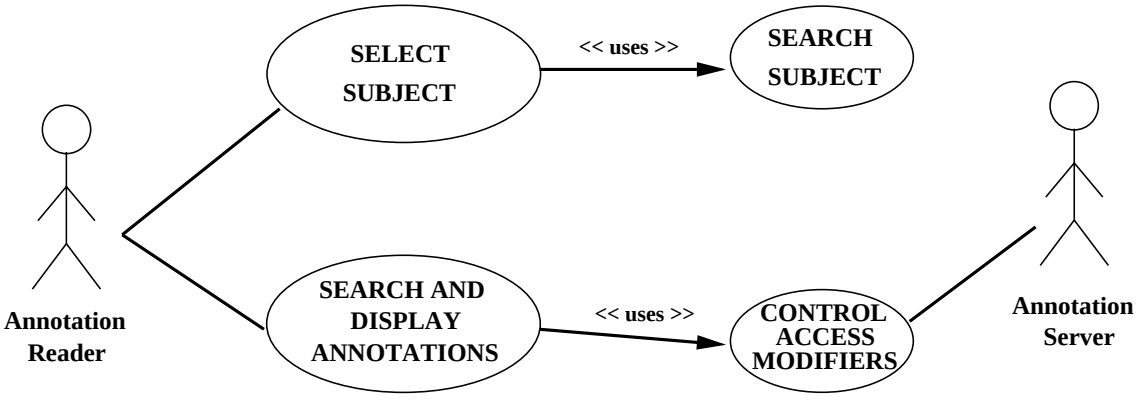

Figura B.6: Visualizzazione delle annotazioni con un soggetto specificato.

- **Visualizzazione delle annotazioni di un determinato progetto**

L'utente può ricercare tutte le annotazioni che si riferiscono ad un determinato progetto che appartiene alla gerarchia dei progetti esistente nel sistema. In tal caso egli seleziona il progetto ed il sistema, dopo opportuni controlli, rende disponibili le annotazioni individuate. Tale funzionalità è riportata nella figura B.7.

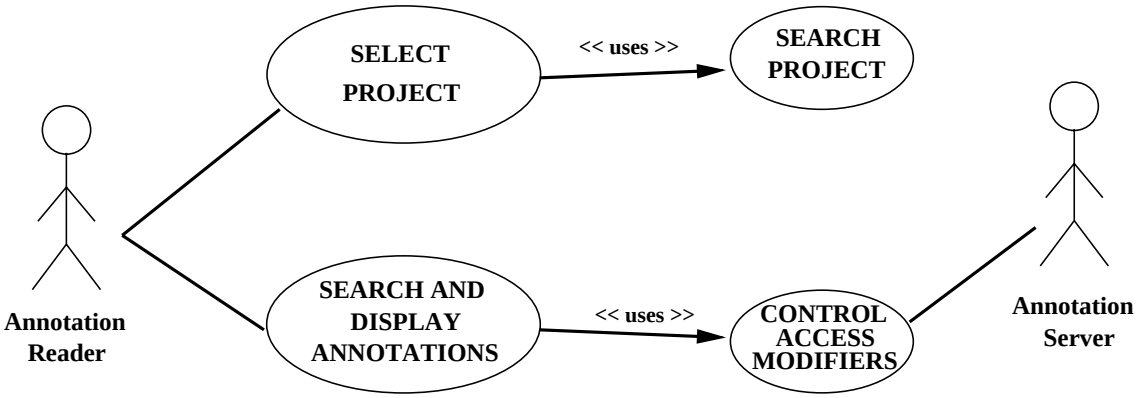

Figura B.7: Visualizzazione delle annotazioni di un determinato progetto.

- **Visualizzazione delle annotazioni di una determinata persona**

  Il sistema offre la possibilità di visualizzare l'insieme delle annotazioni che sono state effettuate da una determinata persona e che l'utente ha il diritto di leggere. Il sistema effettua i controlli necessari e visualizza l'insieme delle annotazioni selezionate (figura B.8).

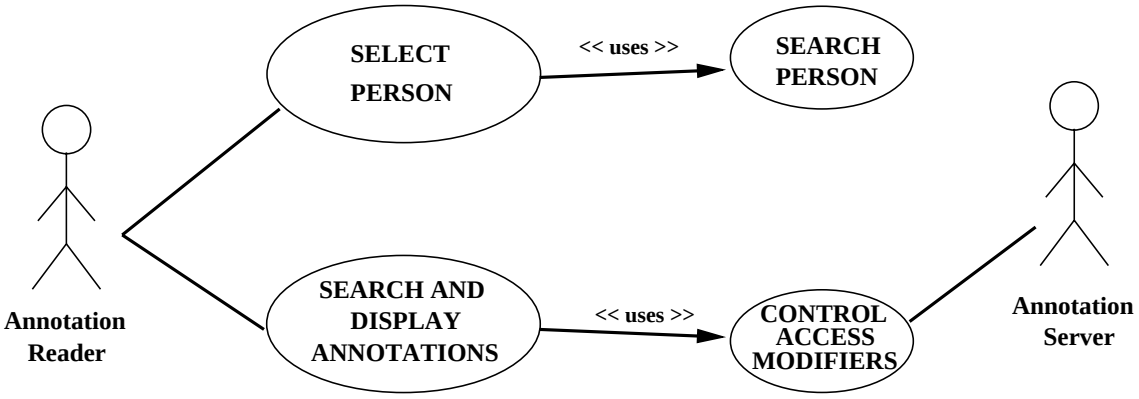

Figura B.8: Visualizzazione delle annotazioni di una determinata persona.

- **Cancellazione di un'annotazione**

  L'*Annotation Server* è in grado di attivare un meccanismo di notificazione anche nel momento in cui un'annotazione viene cancellata. Anzitutto, come si può osservare dalla figura B.9, solo il *System Administrator* può effettuare la cancellazione di un'annotazione.

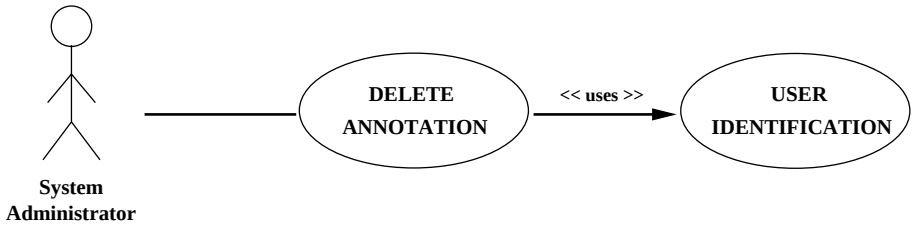

Figura B.9: Cancellazione di un'annotazione.

La figura B.10 mostra in dettaglio i passi che vengono compiuti dal sistema nel momento in cui viene eseguita la procedura di cancellazione.

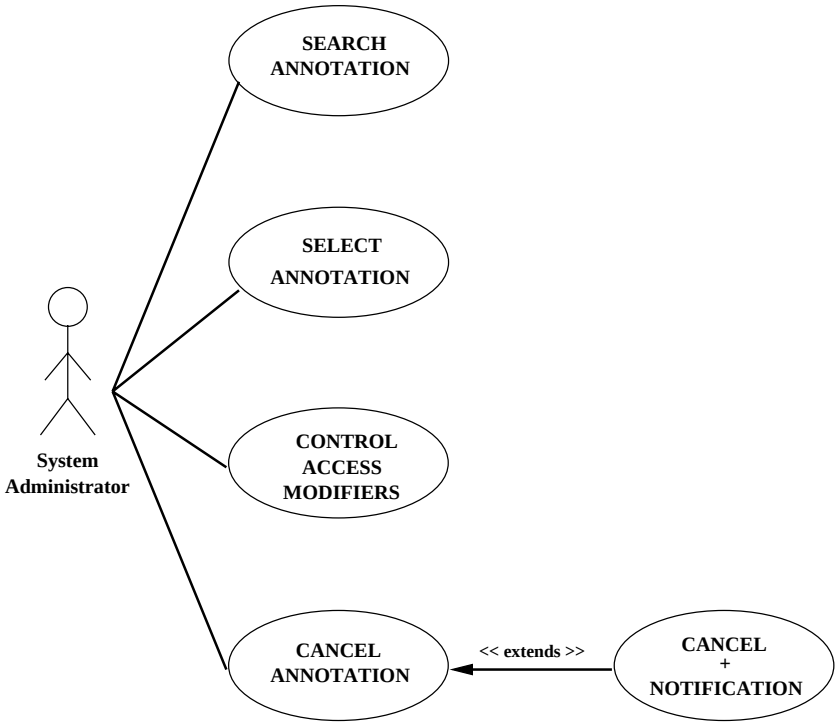

Figura B.10: Cancellazione di un'annotazione -Dettagli-.

A seguito di cancellazione viene attivato il meccanismo di notificazione in base ai criteri descritti nella sezione 5.8.3. La notificazione avviene, come nel caso di creazione di una nuova annotazione e nel caso di inserimento di un utente in un gruppo di progetto, attraverso la selezione degli utenti interessati e la spedizione di una *e-mail*. La cancellazione di un'annotazione è un'operazione estremamente delicata e che quindi necessita qualche osservazione aggiuntiva. Nel momento in cui un'annotazione viene cancellata si pone il grosso quesito di quale dovrà essere il destino dell'eventuale gerarchia di annotazioni sottostante. Purtroppo una soluzione univoca a tale problema non esiste in quanto le decisioni possibili sono diverse a seconda della semantica delle annotazioni sottostanti. Una descrizione dettagliata del problema relativo alla cancellazione è riportata nella sezione 5.8.3. In

questo momento ciò che è necessario puntualizzare è che la cancellazione di annotazioni deve essere un'operazione effettuata con una frequenza molto bassa in quanto determina l'automatica eliminazione di tutte le annotazioni che eventualmente si riferiscono a quella originale. Per tale motivo è stata fatta la scelta di dare la possibilità di cancellare annotazioni solo al *System Administrator* il quale dovrà valutare con cura le situazioni in cui tale operazione è opportuna.

- **Cancellazione delle annotazioni riferite ad un certo progetto/soggetto**

  La figura B.11 mostra in generale la funzionalità di cancellazione delle annotazioni riferite ad un certo progetto o soggetto; questo può essere considerato un caso particolare del più generico *use case* mostrato nella figura B.9.

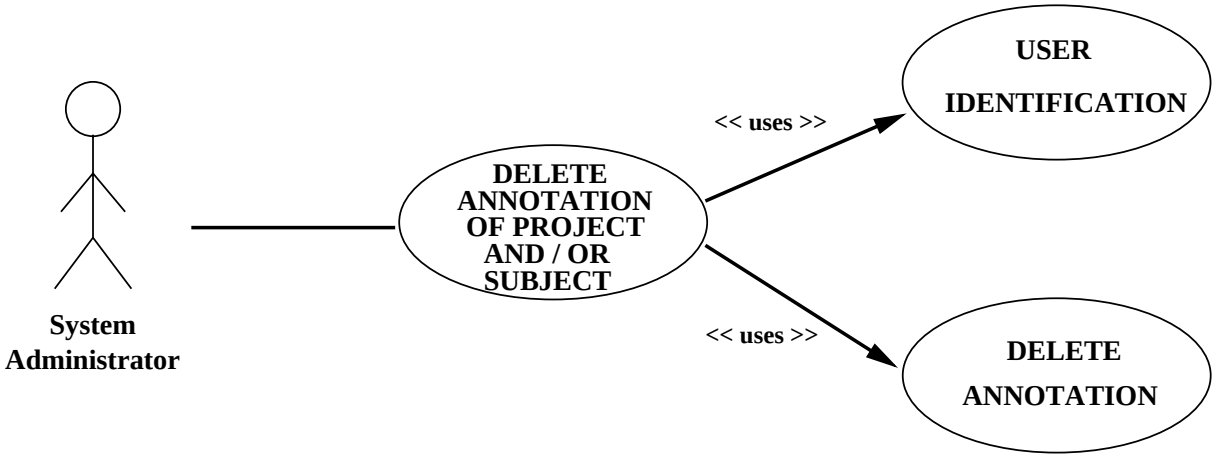

Figura B.11: Cancellazione delle annotazioni riferite ad un certo progetto/soggetto.

La figura B.12 mostra più in dettaglio la funzionalità formalizzata nella figura B.11.

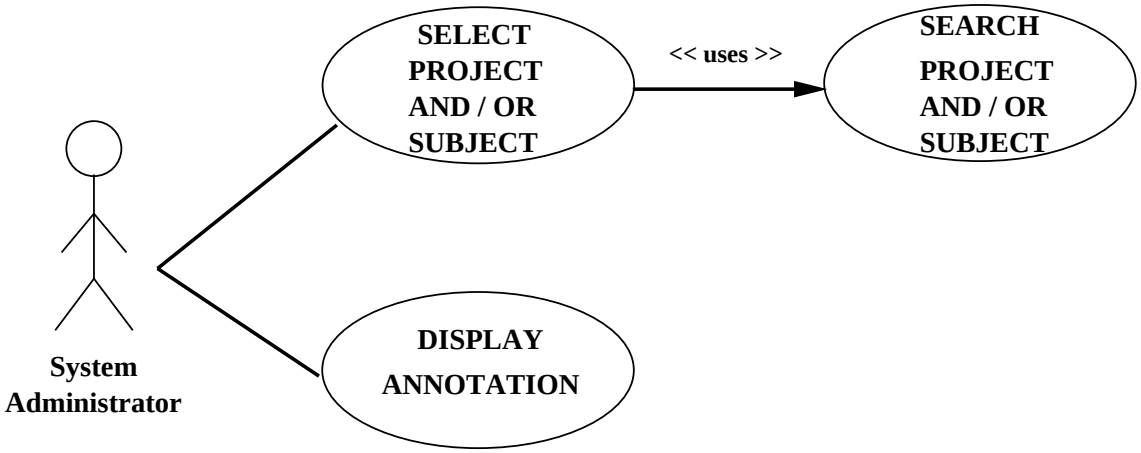

Figura B.12: Cancellazione delle annotazioni riferite ad un certo progetto/soggetto -Dettagli-.

- **Definizione di un nuovo progetto**

  Ogni utente registrato nel sistema deve avere la possibilità di definire un nuovo progetto che si andrà a collocare in una determinata posizione nell'albero gerarchico dei progetti. Ogni progetto deve avere un *Project Administrator* ed un *Subject* a cui si riferisce. Una volta che si è creato il progetto è possibile aggiungere membri ed eventualmente soggetti di interesse.

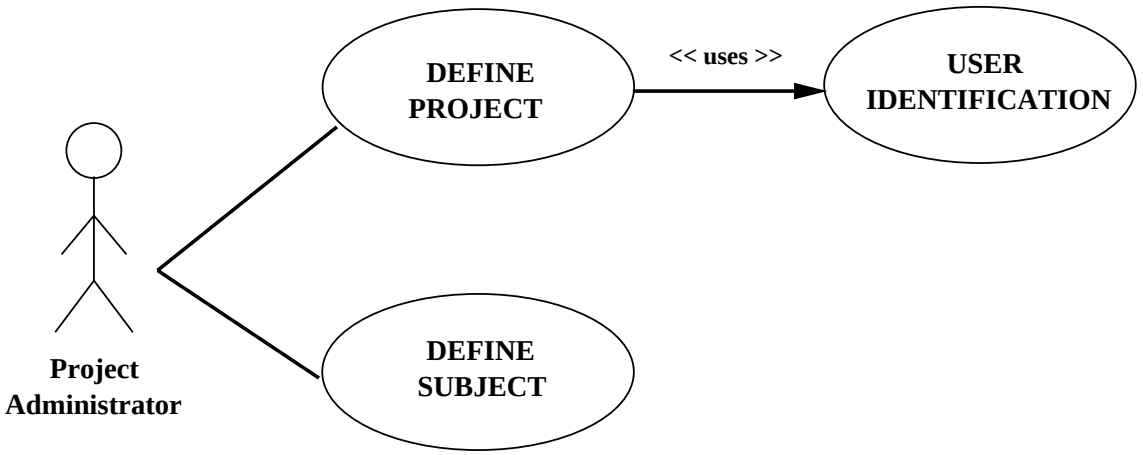

Figura B.13: Definizione di un nuovo progetto.

- **Chiusura di un progetto**

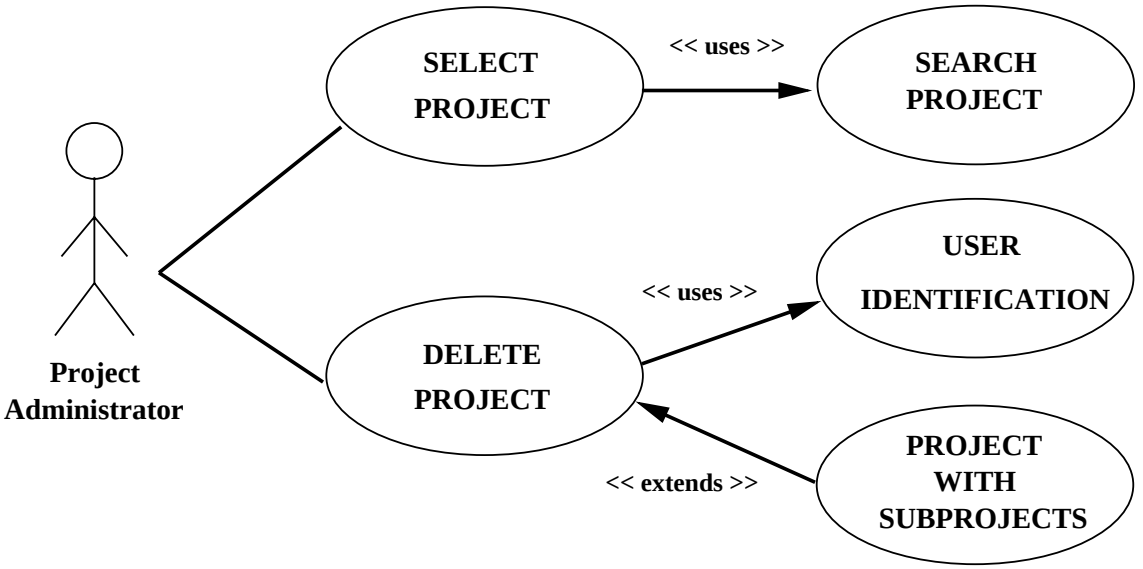

Figura B.14: Chiusura di un progetto.

Il *Project Administrator* ha la possibilità di chiudere il progetto di cui è amministratore; il sistema rende possibile tale operazione solo nel caso in cui il progetto che deve essere

eliminato non possiede alcun sottoprogetto. In caso contrario occorre prima eliminare i sottoprogetti e quindi cancellare il progetto desiderato. Quando un progetto viene cancellato tutti i legami tra soggetti e progetto e tra persone e progetto vengono cancellati. Per quanto riguarda le annotazioni l'*Annotation Server* ridefinisce automaticamente gli attributi *belongs* e *relevants* in base ad una semantica predefinita in funzione del tipo di annotazione che può essere *Private* oppure *Project(P)*. Nel caso in cui l'annotazione è *Private* l'attributo *belongs* assume il riferimento del *Workspace Public*; nel caso di annotazione *Project(P)* invece l'attributo *belongs* assume il riferimento contenuto in *relevants*. Nel caso in cui l'annotazione sia rilevante nell'ambito del progetto cancellato gli attributi *belongs* e *relevants* vengono entrambi riferiti al superprogetto del progetto cancellato.

- **Identificazione di un utente presso il sistema**

  La figura B.15 mostra la procedura attraverso la quale un utente si identifica all'*Annotation Server* prima di compiere una qualsiasi operazione.

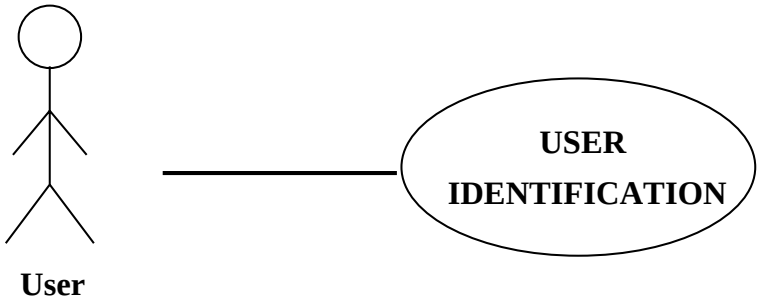

Figura B.15: Identificazione di un utente presso il sistema.

- **Aggiunta di un utente ad un progetto**

  La figura B.16 mostra in generale la procedura di aggiunta di un utente ad un determinato progetto. Tale funzionalità attiva inoltre il meccanismo di notificazione sull'*Annotation Server* il quale compone e spedisce *emails* a tutti i componeti del progetto, che vengono così informati circa l'aggiunta del nuovo membro al gruppo di progetto.

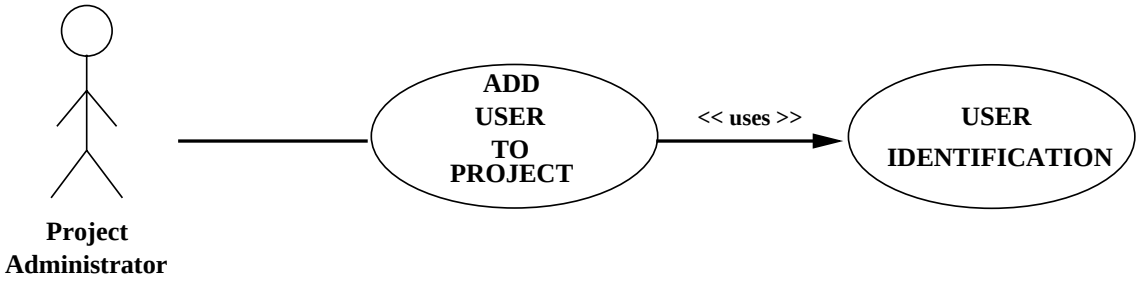

Figura B.16: Aggiunta di un utente ad un progetto.

Tale operazione può essere effettuata solamente dal *Project Administrator* il quale ha la possibilità di aggiungere nuovi membri al gruppo di progetto che amministra. La figu-

ra B.17 mostra in dettaglio i passi che devono essere compiuti per aggiungere un nuovo membro al gruppo di progetto.

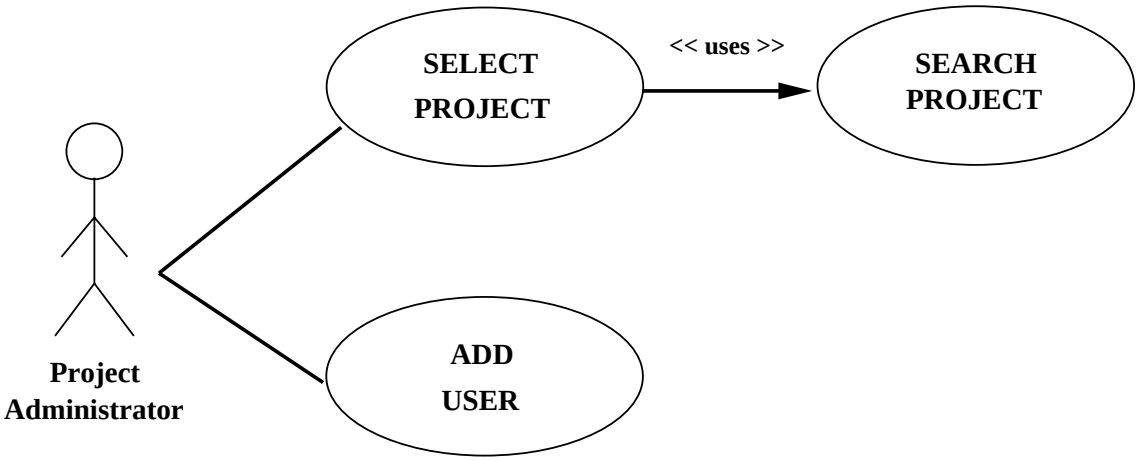

Figura B.17: Aggiunta di un utente ad un progetto -Dettagli-.

- **Cancellazione di un utente da un progetto**

  La figura B.18 mostra la funzionalità offerta dal sistema attraverso la quale il *Project Administrator* può eliminare dei membri dal gruppo di progetto da lui amministrato. Anche tale operazione attiva la procedura di notificazione eseguita dall'*Annotation Server*.

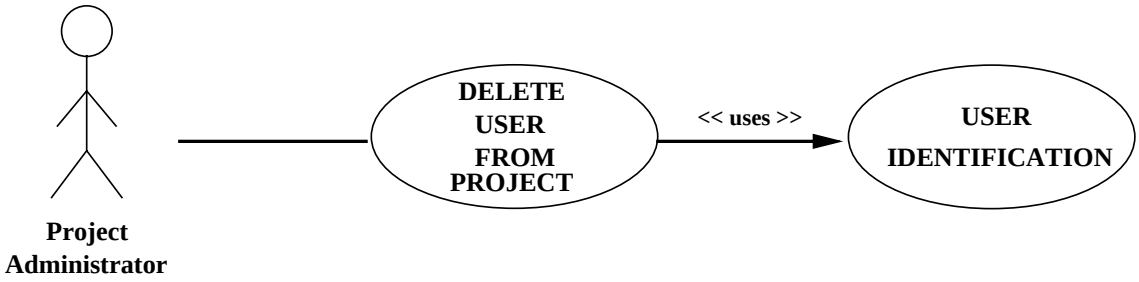

Figura B.18: Cancellazione di un utente da un progetto.

Ciò non significa che l'utente è eliminato dal sistema in quanto comunque egli, sebbene sia stato eliminato dal gruppo di un determinato progetto, continua ad appartenere al *Workspace Public*. Inoltre occorre puntualizzare il fatto che un utente può essere cancellato dal gruppo di progetto solo nel momento in cui non ha un *Login* aperto su quel progetto; in caso contrario occorre prima attendere il *Logout* dell'utente e quindi procedere alla sua cancellazione. Maggiori informazioni a tal proposito sono riportate nel capitolo 7.

La figura B.19 mostra più in dettaglio la procedura di cancellazione sopra discussa.

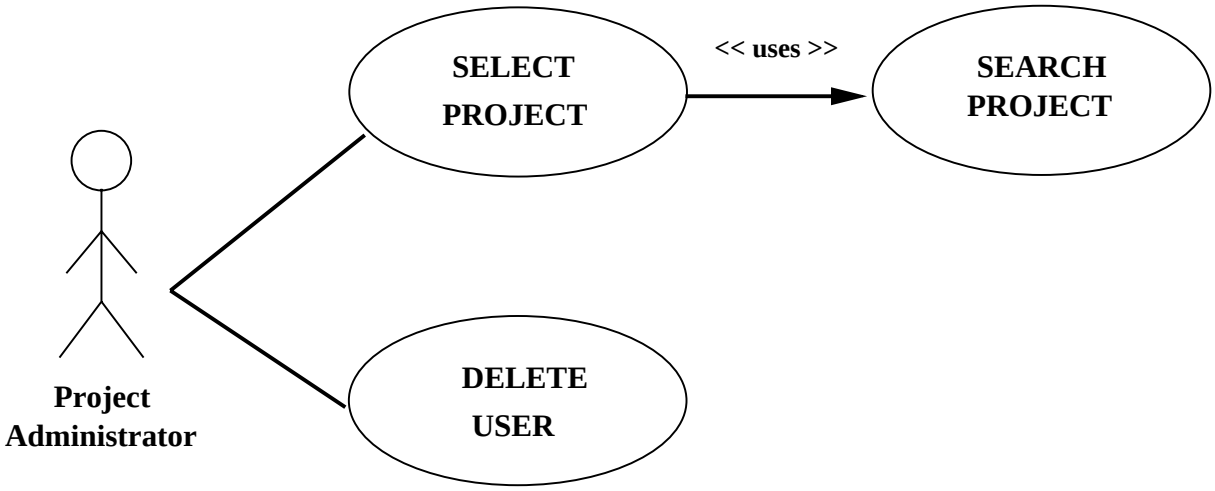

Figura B.19: Cancellazione di un utente da un progetto -Dettagli-.

- **Registrazione di un utente**

  Prima di poter eseguire una qualunque operazione ogni utente deve essere stato precedentemente registrato presso il sistema. Nel momento in cui l'utente viene registrato automaticamente diviene membro del *Workspace Public*. La figura B.20 mostra in generale la procedura di registrazione di un utente. Come si può notare, e come è ovvio che sia, tale funzionalità non richiede alcuna identificazione preliminare in quanto l'utente non è ancora stato registrato.

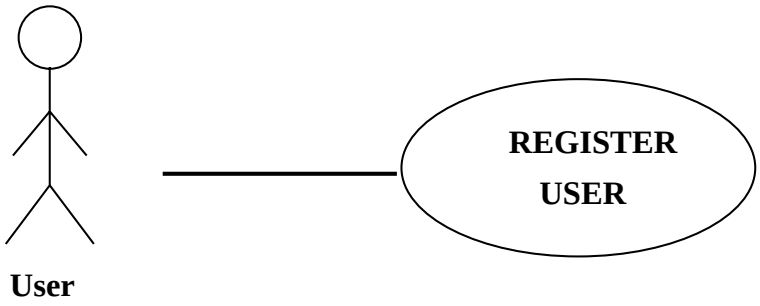

Figura B.20: Registrazione di un utente.

Nella figura B.21 sono riportati in dettaglio i passi che vengono eseguiti durante la procedura di registrazione di un utente presso il sistema. L'utente deve specificare *userName* + *userPassword* che poi dovrà utilizzare ogni volta che effettua un *Login* con il sistema. Il nome dell'utente deve essere univoco nell'ambito dell'intera base di dati e non può essere successivamente modificato dall'utente. La *password* invece può essere successivamente modificata, ovviamente dall'utente stesso che l'ha definita precedentemente.

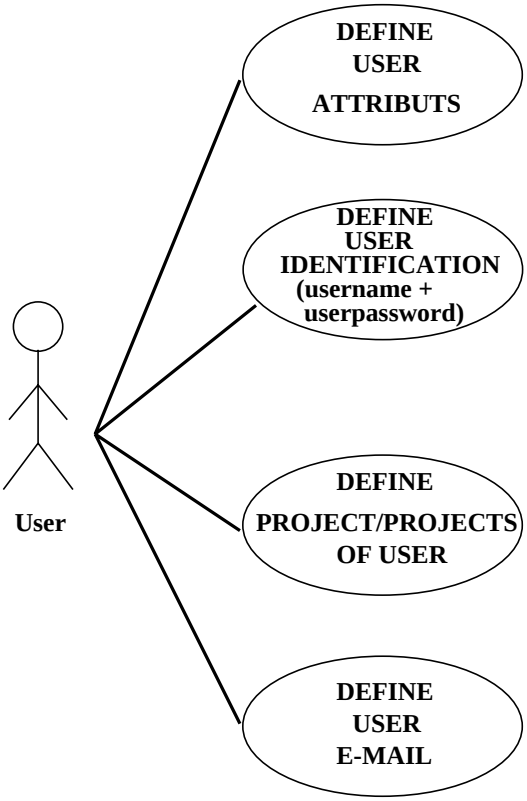

Figura B.21: Registrazione di un utente -Dettagli-.

- **Modifica del profilo utente**

  Ogni utente, previa autentificazione, ha la possibilità di modificare gli attributi che lo caratterizzano quali il nome, la *password* e l'indirizzo di posta elettronica. La figura B.22 mostra tale funzionalità offerta dal sistema.

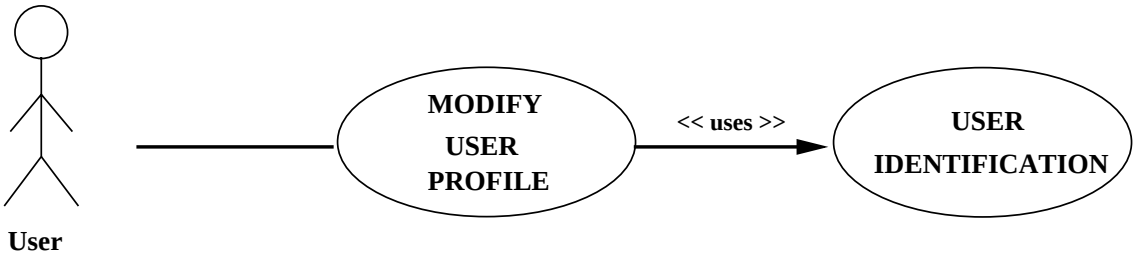

Figura B.22: Modifica del profilo utente.

La figura B.23 riporta i dettagli dell'operazione di modifica del profilo utente; ogni utente ha la possibilità di modificare tutti gli attributi che lo caratterizzano (quali ad esempio la *password* e il suo indirizzo di posta elettronica) fatta eccezione per l'*username*.

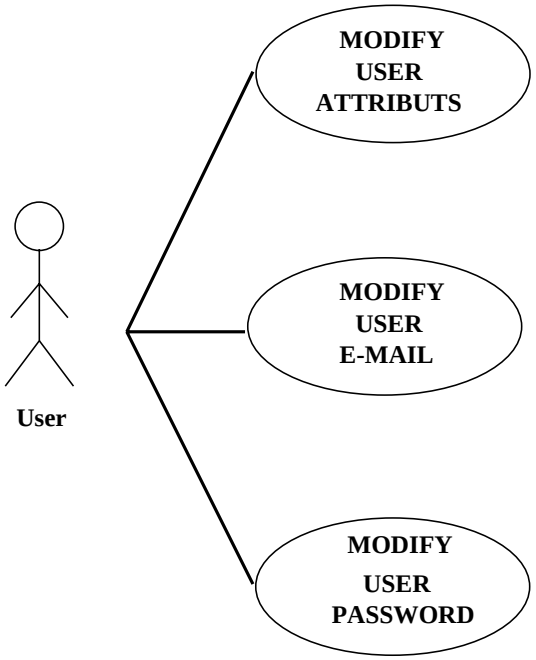

Figura B.23: Modifica del profilo utente -Dettagli-.

- **Gestione dei soggetti di interesse**

  La figura B.24 mostra, da un punto di vista generico, la funzionalità di gestione dei soggetti a cui un determinato utente è interessato. Con il termine gestione si intendono sostanzialmente le due operazioni di aggiunta di nuovi soggetti e di cancellazione di soggetti esistenti.

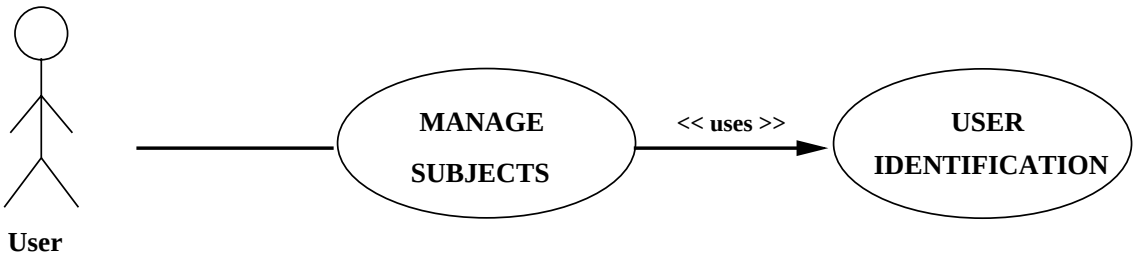

Figura B.24: Gestione dei soggetti di interesse.

La figura B.25 riporta in modo più dettagliato l'operazione di aggiunta di nuovi soggetti; da notare la possibilità offerta all'utente di specificare soggetti in base ad una determinata modalità che può essere *Public* oppure *No Public*. Per maggiori dettagli a tal riguardo fare riferimento al capitolo 5 in cui è riportata la descrizione del modello sviluppato ed al capitolo 8 in cui sono riportati i dettagli relativi all'implementazione del prototipo.

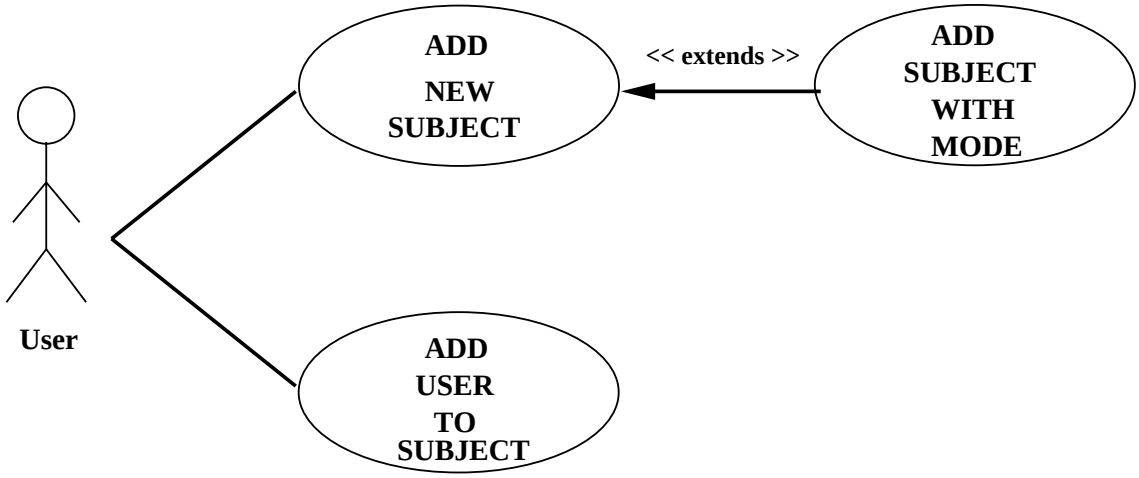

Figura B.25: Aggiunta di nuovi soggetti.

La figura B.26 riporta dettagliatamente i passi dell'operazione di cancellazione di soggetti a cui un utente non è più interessato. Tale operazione non comporta ovviamente la cancellazione dei soggetti dalla base di dati[2] bensì la cancellazione dei loro legami con l'utente in questione.

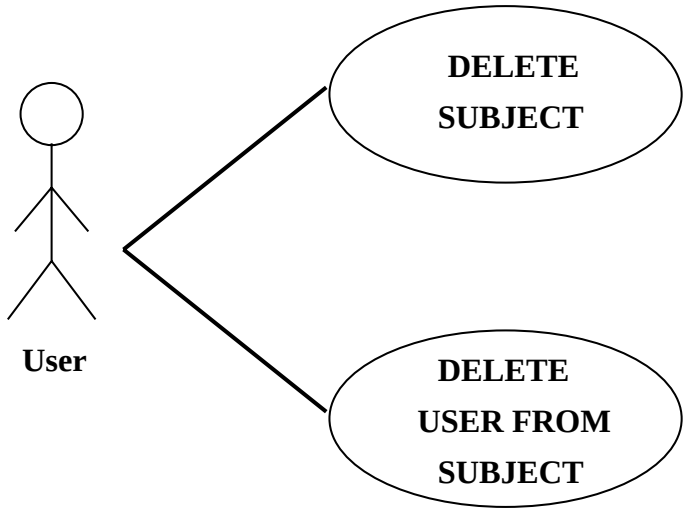

Figura B.26: Cancellazione di soggetti esistenti.

- **Notificazione attraverso *e-mails***

  Il meccanismo di notificazione è realizzato mediante l'utilizzo della posta elettronica; in tal caso si parla anche di notificazione *off-lines*. Si possono individuare tre situazioni in cui occorre effettuare una notifica tramite *e-mail* agli utenti interessati:

[2]Questo perché potrebbero esserci altri utenti, annotazioni oppure progetti che fanno riferimento a quel determinato soggetto.

- Notifica a seguito di nuova annotazione
- Notifica a seguito di cancellazione
- Notifica a seguito di inserimento o cancellazione di un utente da un gruppo di progetto

Ciò significa che il meccanismo di notificazione deve essere attivato ogni volta che un qualsiasi utente crea un'annotazione oppure quando l'amministratore del sistema effettua una cancellazione oppure quando un amministratore di progetto inserisce od elimina un nuovo membro dal gruppo di progetto. Le regole in base alle quali è realizzato l'algoritmo di notifica sono esposte in maniera esaustiva nella sezione 5.8.2. La figura B.27 mostra in generale la funzionalità di notificazione realizzata dall'*Annotation Server*.

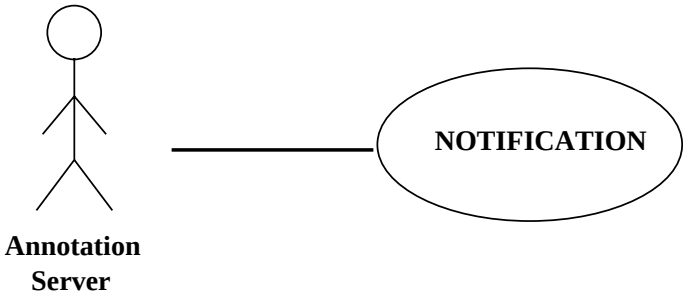

Figura B.27: Meccanismo di notificazione.

La figura B.28 mostra in dettaglio la procedura di notificazione.

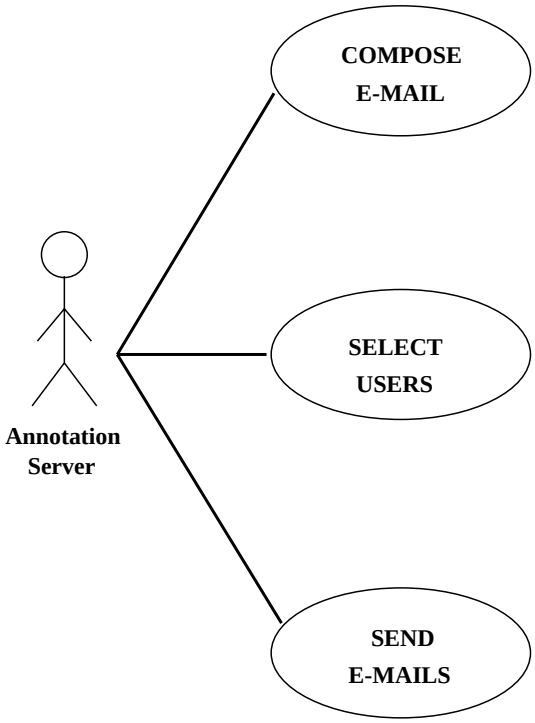

Figura B.28: Meccanismo di notificazione -Dettagli-.

In tale *use case* l'attore è rappresentato dall'*Annotation Server*[3] che è un componente dell'architettura del sistema descritta nella sezione 7.1.

Ogni volta che viene effettuata un'operazione che richiede la procedura di notificazione (creazione di una nuova annotazione, cancellazione di un'annotazione già esistente oppure inserimento/cancellazione di un utente dal gruppo di progetto) l'*Annotation Server* in primo luogo determina quali utenti, registrati nel sistema, dovranno ricevere la notifica. Quindi l'*Annotation Server* compone una *e-mail* da spedire ad ogni utente opportunamente selezionato. L'*e-mail* contiene tutte le informazioni necessarie per comprendere cosa sia accaduto nel sistema.

La notificazione è un aspetto molto importante dell'*Annotation Server* che lo rende un componente attivo, capace cioè di reagire a degli eventi, eseguendo delle azioni nel caso in cui determinate condizioni siano verificate.

B.2 Il Modello

In tale sezione è riportato il *Class Diagram*, dettagliato in base alla sintassi UML, del modello proposto.

La figura B.29 fa riferimento a quanto riportato nella sezione 5.3 in cui è possibile ritrovare anche uno schema semplificato avente lo scopo di consentire la comprensione dei concetti e dei meccanismi di base. Nel diagramma riportato di seguito sono specificate le regole di navigabilità, le regole di visibilità, gli attributi ed i metodi più importanti per ogni classe facente parte del modello[4].

Il modello descritto nella figura B.29, così come quello riportato nella sezione 5.3, non specificano i meccanismi amministrativi che si sono adottati invece a livello di implementazione della base di dati. Tra tali meccanismi sono compresi la tabella KEYS (riportata nella figura 5.20 del capitolo 5) e l'attributo *password* della classe *Person*; tale scelta è stata effettuata in base alla logica secondo cui tali meccanismi di autentificazione rappresentano un aspetto puramente implementativo. Inoltre questi meccanismi devono essere del tutto ortogonali al modello sviluppato ovvero indipendenti dai criteri in base ai quali il sistema è stato realizzato; questo è un aspetto estremamente importante che deve sempre essere preso in considerazione in fase di progettazione al fine di ottenere soluzioni scalabili e soprattutto flessibili[5].

La specifica completa delle interfacce realizzate è riportata nel capitolo 7 di cui si consiglia la lettura ad integrazione del seguente appendice.

[3] L'*Annotation Server* da un punto di vista concettuale può essere considerato correttamente un attore in quanto anch'esso richiede al sistema lo svolgimento di una certa funzionalità che è proprio rappresentata dalla procedura di notificazione.

[4] Al fine della comprensione di tali termini si consiglia la lettura del capitolo 6 e dei riferimenti bibliografici riportati alla fine del testo.

[5] Avere meccanismi di autentificazione ortogonali al modello significa infatti avere la possibilità di modificare la loro logica di funzionamento senza per questo dover portare delle modifiche ai criteri in base ai quali il sistema è stato organizzato.

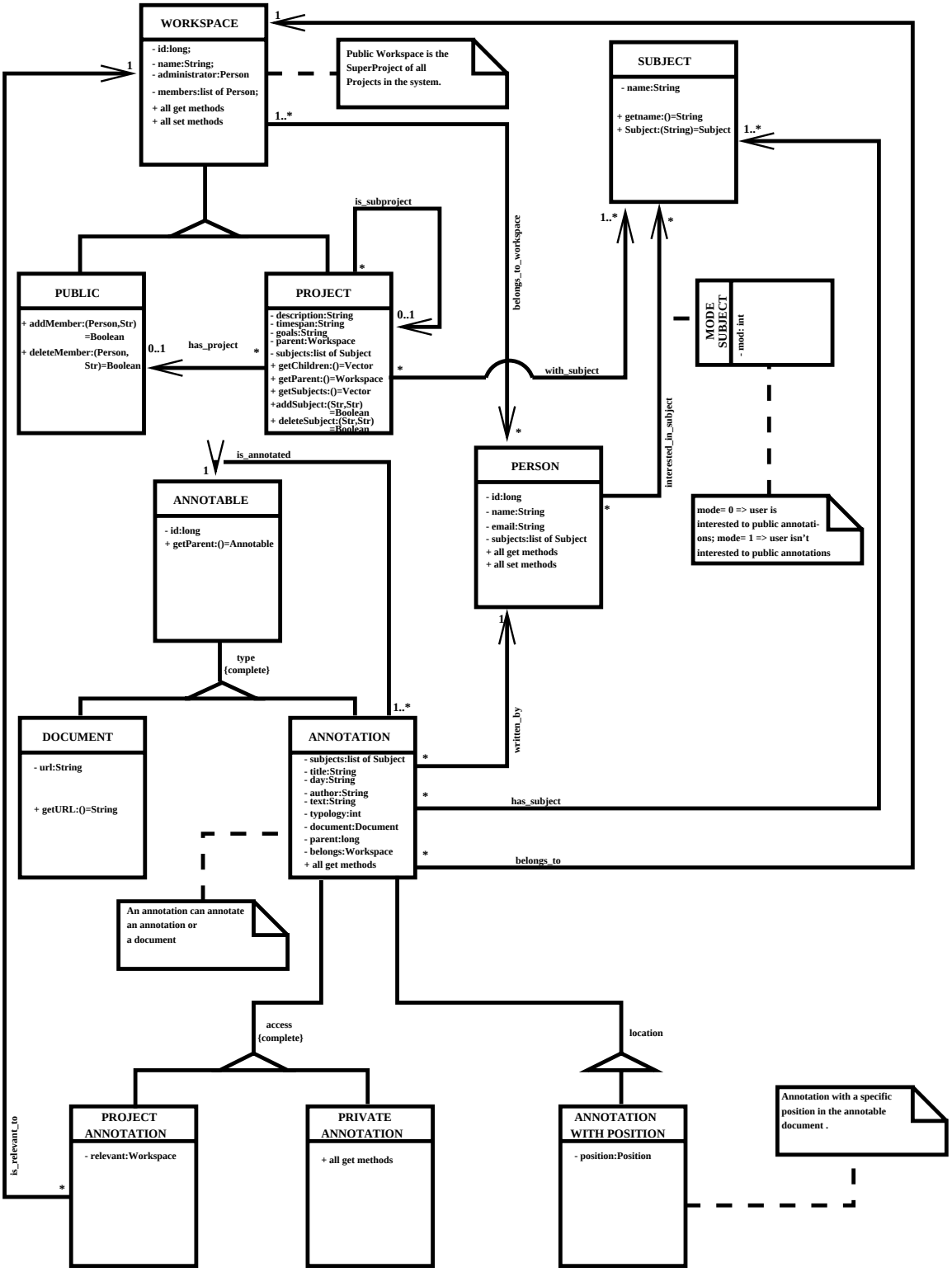

Figura B.29: Modello dettagliato del sistema.

Appendice C

Implementazione

In tale appendice è riportata, a titolo esemplificativo, l'implementazione dell'interfaccia *ModeSubject* la quale è molto semplice da comprendere e presenta anche un metodo (*setMod(int mod,String key)*) che interagisce con la base di dati. La classe *ModeSubject* estende la classe *Subject* da cui eredita metodi ed attributi. Il codice riportato è interamente commentato.

```
//name of the package
package annotationSystem;
//import java Packages
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.util.*;
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;
//class definition
public class ModeSubjectImpl extends annotationSystem.
SubjectImpl implements ModeSubject {
 private int mod;
 //Constructor
 public ModeSubjectImpl(String nameParameter,
int modParameter) throws RemoteException {
    super (nameParameter) ;
    mod=modParameter;

 }//end ModeSubjectImpl

 //get the mode of the Subject
 public int getMod() throws RemoteException{
   return this.mod;
 }//end getMod
```

```
 //set the mode of the subject
 public int setMod(int mod) throws RemoteException{
    this.mod=mod;
    return this.mod;
 }//end setMod

 //interaction with the database
 public boolean setMod(int mod,String key)
throws RemoteException{
   boolean out=false;
      try{
      // load drivers
      Class.forName("oracle.jdbc.driver.OracleDriver");
      // make a connection with the database
      Connection conn = DriverManager.getConnection("jdbc:
oracle:thin:sts04/sts04@hp00.rz.tu-harburg.de:1521:oracle");
      System.out.println(">>....Create Connection....");
      // check for SQLWarnings
      checkForWarning(conn.getWarnings());
        //create PreparedStatements
      PreparedStatement pstmt=conn.prepareStatement("SELECT
idPerson FROM KEYS WHERE key=?");
      pstmt.setString(1,key);
      ResultSet rs=pstmt.executeQuery();
      if (rs.next()){
         pstmt=conn.prepareStatement("UPDATE SUBJECTOFPERSON
SET mod=? WHERE (subjectName=?) AND (idPerson=?)");
        pstmt.setInt(1,mod);
        pstmt.setString(2,this.getName());
        pstmt.setLong(3,rs.getLong(1));
        int row=pstmt.executeUpdate();
        if (row>0){
          this.mod=mod;
          out=true;
}//end if
       // close Statemts
       pstmt.close();
       // release the connection with database
       conn.close();
     }//end if
    }//end try
    catch(SQLException e){
```

```
      System.out.println("//  SQL-EXCEPTION //");
      System.out.println("SQLstate:"+ e.getSQLState());
      System.out.println("Message:"+ e.getMessage());
      System.out.println("Vendor:" + e.getErrorCode());
      System.out.println("");
    }//end catch
    catch(Exception e){
      System.out.println("// Exception//");
      System.out.println(e.getMessage());
      e.printStackTrace();
    }//end catch
    return out;
  }//end setMod
   //check the warnings
   public static boolean checkForWarning(SQLWarning w)
throws SQLException{
    boolean rt = false;
    if (w != null){
      rt =true;
      System.out.println("//SQLWarnings//");
    }//end if
    while (w != null){
      System.out.println("SQLstate:"+ w.getSQLState());
      System.out.println("Message:"+ w.getMessage());
      System.out.println("Vendor:" + w.getErrorCode());
      System.out.println("");
      w=w.getNextWarning();
    }//end while
    return rt;
  }//end checkForWarning

  public static void main(String args[]){
    // create and install a security manager
    System.setSecurityManager(new RMISecurityManager());
    try{
       //reference for the remote object
       ModeSubjectImpl obj = new ModeSubjectImpl
("ModeSubjectServer",0);
       Naming.rebind("//www.sts.tu-harburg.de/"
    +obj.getName(),obj);
       System.out.println(obj.getName()
 +" bound in registry");
    }//end try
```

```
    catch (Exception e){
       System.out.println("SubjectImpl error: "
       +e.getMessage());
      e.printStackTrace();
    }//end catch
  }//end main
}//end class ModeSubjectImpl
```

È molto interessante analizzare un po' più in dettaglio il metodo *main()* che crea il riferimento sull'*Annotation Server* affinché l'oggetto remoto *ModeSubjectImpl* possa essere utilizzato dal *Proxy Server*; tutte le classi sono state realizzate sull' *Annotation Server* seguendo tale principio. Meritano attenzione inoltre le modalità in base alle quali sono stati gestiti gli errori attraverso l'implementazione di codice ad hoc ed utilizzando il meccanismo delle eccezioni messo a disposizione dal linguaggio Java.

Ultima considerazione riguarda il fatto che nell'implementazione sopra riportata sono stati definiti esplicitamente i parametri di configurazione tecnica, quali il nome del *Driver* per accedere alla base di dati, il nome della base di dati e così via. Le classi che costituiscono il prototipo effettivo invece fanno uso dei parametri tecnici riportati nel file *AnnotationSystemConfiguration.txt* come spiegato nel capitolo 8.

Bibliografia

Andreessen 1993: Marc Andreessen. *Notification and Monitoring,Suggests that annotation servers do filtering*. http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Notes/Notification.html, 1993.

Beck 1989: Kent Beck. *A Laboratory For Teaching Object-Oriented Thinking*. SIGP LAN, 1989.

Berners-Lee and Connolly 1993: Tim Berners-Lee and Daniel Connolly. Hypertext markup language: A representation of textual information and metainformation for retrieval and interchange. Technical report, CERN and Atrium Technology Inc., July 1993.

Booch and Rumbaugh 1995: Grady Booch and James Rumbaugh. *Unified Method for Object-Oriented Development Version 0.8*. Rational Software Corporation, 1995.

Booch and Rumbaugh 1997: Grady Booch and James Rumbaugh. *Unified Method for Object-Oriented Development Version 1.0*. Rational Software Corporation, 1997.

Booch 1994: Grady Booch. *Objektorientierte Analyse und Design*. Addison-Wesley, 1994.

Braverman 1996: Alan Braverman. *Group and Public Annotation Protocol*. http://www.ncsa.-uiuc.edu/SDG/Software/XMosaic/Annotations/protocol.html, 1996.

Brockschmidt 1995: Kraig Brockschmidt. Design considerations for implementing a simple OLE automation controller. *Microsoft Systems Journal*, 10(5):57–??, May 1995.

Brooks et al. *1995:* Charles Brooks, Murray S. Mazer, Scott Meeks, and Jim Miller. Application specific proxy servers aas HTTP stream transducers. In *Proceedings of the 4th International World Wide Web Conference*, December 1995.

Bush 1945: Vannevar Bush. *As We May Think*. Atlantic, August 1945. Quoted in Chapter 3, [Rheingold 91].

Ceri and Ramakrishnan 1996: Stefano Ceri and Raghu Ramakrishnan. Rules in database systems. *ACM Computing Surveys*, 28(1):109–111, March 1996.

Ceri et al. *1997:* Stefano Ceri, Carlo Zaniolo, Christos Faloutsos, Richard T. Snodgrass, V. S. Subrahmanian, and Roberto Zicari. *Introduction to Advanced Database Systems*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1997.

Corporation 1992a: Oracle Corporation. *Oracle 7 Server Concepts Manual. Part nummer 6693-70.* Redwood City, California, 1992.

Corporation 1992b: Oracle Corporation. *Oracle 7 Server Sql Language Reference Manual. Part nummer 778-70.* Redwood City, California, 1992.

Davis and Huttenlocher 1995: J.R. Davis and D.P. Huttenlocher. Shared annotation for cooperative learning. In *Proceedings of Computer Support for Cooperative Learning Conference, pp.84-88.* http://www-cscl95.indiana.edu/cscl95/davis.html, 1995.

developers 1993: Mosaic developers. *Group Annotations in NCSA Mosaic*. http://www.ncsa.-uiuc.edu/SDG/Software/Mosaic/Docs/group-annotations./html, 1993.

D.LaLiberte 1997: D.LaLiberte. *HyperNews Homepage.* http://union.ncsa.uiuc.edu/Hyper-News/get/hypernews.html, 1997.

D.LaLiberte 1995: D.LaLiberte. *Collaboration Technologies on th Web*. http://union.ncsa.-uiuc.edu/HyperNews/get/hypernews.html, 1995.

D.LaLiberte 1997: D.LaLiberte. *HyperNews Homepage, A Prototype of our annotationsystem.* http://www.ncsa.uiuc.edu/HyperNews/get/hypernews.html, 1997.

Duffy 1993: Thomas M. Duffy. ”hypertext: The convergence of contemporary critical theory and technology,” by george P. landow. *Hypermedia*, 5(1):74–77, 1993.

Fish et al. *1988:* Robert S. Fish, Robert E. Kraut, and Mary D. P. Leland. Quilt: A collaborative tool for cooperative writing. In *Proceedings of the Conference on Office Automation Systems*, Collaborative Work, pages 30–37, 1988.

Fountain et al. *1990:* Andrew M. Fountain, Wendy Hall, Ian Heath, and Hugh C. Davis. MI-CROCOSM: An open model for hypermedia with dynamic linking. In *Proceedings of the ECHT’90 European Conference on Hypertext*, Building Hypertext Applications, pages 298–311, 1990.

Fowler 1997: Martin Fowler. *UML distilled*. Addison-Wesley, 1997.

Gilb and Graham 1993: Tom Gilb and Dorothy Graham. *Software Inspection*. Addison-Wesley, 1993.

Glouberman 1996: Misha Glouberman. *Adding Comments to the Web*. First Public Draft misha@the-wire.com July 26, 1996.

GmbH 1997: HyperWave Information Management GmbH. *HyperWave Homepage*. http://-www.hyperwave.de/, 1997.

Gramlich 1994: Wayne C. Gramlich. *Annotation System Isuuses*. http://playground.sun.com-/ gramlich/1994/annote/issues/index.html, 1994.

Hahn and Stout 1994: Harley Hahn and Rick Stout. *The Internet Complete Reference*. Osborne/McGraw-Hill, Berkeley, CA, USA, 1994. *Highly recommended.* This is by far the most complete, illuminating, useful, exciting, readable :-) book on the Internet .... Covering `mail`, `telnet`, `finger`, `usenet`, `ftp`, `talk`, IRC, `gopher`, Wais, WWW, etc., ... And there is also a Catalog of Internet Resources (over 140 pages) ....

J.Davis 1996: J.Davis. *CoNote (Annotation) Home Page.* http://dri.cornell.edu.pub/davis/-annotation.html, 1996.

Knuth 1973: Donald E. Knuth. *The Art of Computer Programming: Fundamental Algorithms*, volume 1. Atlantic, 2 edition, 1973.

Konsortium 1996: Medoc Konsortium. *Ariadne Homepage*. http://ariadne.inf.fu-berlin.de:-8000/, 1996.

Koszarek et al. *1990:* J. L. Koszarek, T. L. Lindstrom, J. R. Ensor, and S. R. Ahuja. A multi-user document review tool. In S Gibbs and A. A. Verrighn-Stuart, editors, *Proceedings of IFIP WG8.4 Conference on Multi-User Interfaces and Applications*, Crete, 1990. North Holland.

Lawton and Smith 1993: Daryl T. Lawton and Ian E. Smith. The knowledge weasel hypermedia annotation system. In *Proceedings of ACM Hypertext'93*, Papers, pages 106–117, 1993.

Lesk 1989: Michael Lesk. What to do when there's too much information. In *ACM Hypertext'89 Proceedings*, Information Retrieval II, pages 305–318, 1989.

Lesk 1997: Michael Lesk. *Practical Digital Libraries: Books, Bytes, and Bucks.* Morgan Kaufmann Publishers, 2929 Campus Drive, Suite 260, San Mateo, CA 94403, USA, 1997.

Levy and Marshall 1995: David M. Levy and Catherine C. Marshall. Going digital: A look at assumptions underlying digital libraries. *Communications of the ACM*, 38(4):77–84, April 1995.

Li et al. *1996:* Wei Li, Susan Gauch, John Gauch, and Kok Meng Pua. VISION: A digital video library. In *Proceedings of DL'96*, 1996.

Marshall 1997: Catherine C. Marshall. Annotation: from paper books to the digital library. In Robert B. Allen and Edie Rasmussen, editors, *Proceedings of the 2nd ACM International Conference on Digital Libraries*, pages 131–141, New York, July23–26 1997. ACM Press.

M.B.Davidson et al. *1995:* M.B.Davidson, M.S.Mazer, and Charles Brooks. *Dynamic Integration of HTTP Stream Transforming Services.* OSF Research Institute technical report, Cambridge, MA, USA, February 1995.

Miksa and Doty 1994: Francis Miksa and Philip Doty. Intellectual realities and the digital library. In *Digital Libraries '94 Proceedings*, 1994.

MotherJones 1997: MotherJones. *Live Wire*. http://www.motherjones.com/, 1997.

Moulthrop 1992: Stuart Moulthrop. Toward a rhetoric of information texts. In *Proceedings of the Fourth ACM Conference on Hypertext*, Hypertext and the Mind, pages 171–180, 1992.

Nickerson 1992: G. Nickerson. WorldWide Web: hypertext from CERN. *Computers in Libraries*, 12(11):75–77, December 1992.

Niederée et al. *1996:* C. Niederée, C. Hattendorf, S. Müßig, M. Warnke, and J. W. Schmidt. Warburg electronic library: Eine digitale Bibliothek für die Politische Ikonographie. *unihh forschung*, XXXI, 1996.

O'Hara 1997: Kenton O'Hara. A comparison of reading paper and on-line documents. In *In Proceedings of CHI '97,March 22-27*, 1997.

Phelps and Wilensky 1996: Thomas A. Phelps and Robert Wilensky. Toward active, extensible, networked documents: Multivalent architecture and applications. In *Proceedings of DL'96*, 1996.

Prinz and Syri 1997: W. Prinz and A. Syri. Two complementary tools for the cooperation in a ministerial environment. *J.UCS: Journal of Universal Computer Science*, 3(8):843–??, August 1997.

R.Bentley et al. *1995:* R.Bentley, T.Horstmann, K.Sikkel, and J.Trevor. Supporting collaborative information sharing with the world wide web: The bscw shared workspace system. In *Proceedings of the 4th international WWW Conference , Boston,pages 63-74,1995*, 1995.

Röscheisen et al. *1994:* Martin Röscheisen, Christian Mogensen, and Terry Winograd. Shared web annotations as A platform for third-party value-added information providers: Architecture, protocols, and usage examples. Technical report, Computer Science Department, Stanford University, , November 1994.

Röscheisen et al. *1995a:* M. Röscheisen, C. Mogensen, and T. Winograd. Beyond browsing: Shared comments, SOAPs, trails and on-line communities. In *Proceedings of the Fourth World-Wide Web Conference*, pages 739–749, Darmstadt, Germany, 1995.

Röscheisen et al. *1995b:* Martin Röscheisen, Terry Winograd, and Andreas Paepcke. Content ratings and other third-party value-added information: Defining an enabling platform. *CNRI D-Lib Magazine*, August 1995.

Roscheisen et al. *1997:* Martin Roscheisen, Christian Mogensen, and Terry Winograd. Shared web annotations as a platform for third-party value-added, information providers: Architecture, protocols, and usage examples. Technical Report CS-TR-97-1582, Stanford University, Department of Computer Science, January 1997.

S.Ceri et al. *1996:* S.Ceri, P.Atzeni, S.Paraboschi, and R.Torlone. *Basi di dati - concetti, linguaggi ed architetture*. McGraw-Hill, Milano, 1996.

Schmidt et al. *1997:* J. W. Schmidt, G. Schröder, C. Niederée, and F. Matthes. Linguistic and architectural requirements for personalized digital libraries. *International Journal of Digital Libraries*, 1(1), 1997.

Schnase et al. *1994:* John L. Schnase, John J. Leggett, Edward S. Metcalfe, Nancy R. Morin, Edward L. Cunnius, Jonathan S. Turner, Richard K. Furuta, Leland Ellis, Michael S. Pilant, Richard E. Ewing, Scott W. Hassan, and Mark E. Frisse. The colib project–enabling digital botany for the 21st century. In *DL '94 Proceedings*, 1994.

Sedlmayer 1996: G. Sedlmayer. *Integration von Anmerkungsmöglichkeiten in Mehrbenutzereditor IRIS.* TU München, Institut Informatic, Diplomarbeit, 1996.

Swadley 1996: Richard K. Swadley. *Java unleashed*. Sams Publishing, 1996.

Widom and Ceri 1996: John Widom and Stefano Ceri. *Active Database Systems*. Morgan Kaufmann Publishers, San Mateo, California, 1996.

Wirfs-Brock et al. *1990:* Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener. *Designing Object-Oriented Software*. Prentice-Hall, Englewood Cliffs, NJ 07632, 1990.

Woelk and Kim 1987: D. Woelk and W. Kim. Multimedia information management in an object-oriented database system. In *vldb*, pages 319–329, Brighton, England, 1987.