Indice

1 Sistemi di digitalizzazione delle lezioni

1.1 Introduzione

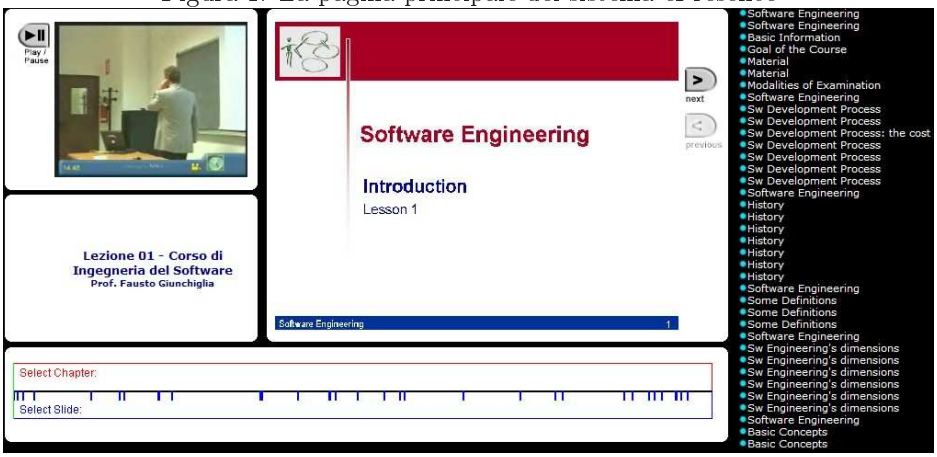
I sistemi di digitalizzazione delle lezioni a basso costo consentono, a qualsiasi studente collegato ad internet, di vedere lezioni, seminari, presentazioni varie stando comodamente seduti davanti al proprio computer. Per fruire di una lezione è sufficiente avere un browser web con alcuni plug-in installati. La lezione può essere navigata interattivamente grazie a diversi strumenti. Infatti sarà possibile andare al punto desiderato della lezione semplicemente cliccando sul titolo desiderato oppure su una timeline. Audio e video possono essere accompagnati dalle immagini che vengono proiettate in classe (se il corso registrato lo richiedere) in modo tale che l'utente possa seguire perfettamente il professore come se fosse in classe.

I vantaggi sono diversi... con questo sistema persone che non possono presenziare alla lezione verranno agevolate, senza parlare di chi ha difficoltà a seguire causa la lingua diversa da quella madre. Solo con questi sistemi è possibile riascoltare la parte della lezione più complessa e difficile da comprendere.

Andiamo ora ad analizzare vari sistemi che permettono di fare questo partendo dal più datato (ePresence) fino ad arrivare alla più recente versione di Lode.

1.2 Il sistema ePresence

Figura 1: La pagina principale del sistema ePresence



ePresence è un sistema sviluppato presso il Knowledge Media Design Institute dell'Università di Toronto dal gruppo dei Proff. Ron Baecker e Gale Moore. Il sistema è stato sperimentato presso il Dipartimento di Informatica e Telecomunicazioni grazie ad un accordo stipulato con il Prof. Baecker.

Per poter utilizzare ePresence è necessario un browser con il plug-in Java e Javascript abilitati. Il video della lezione è in formato Real Media e quindi per poter es-

sere visualizzato correttamente è necessario installare il plug-in apposito (Real Media appunto).

La pagina principale (index.htm) si presenta come in Figura 1 ed è suddivisa in 4 frame ben distinti. A ciascuno di questi è associata una pagina web.

Ecco i frame nel dettaglio:

video_frame: questo frame ingloba sia il video sia le informazioni riguardanti la lezione: nome del corso, numero della lezione e professore che la tiene. Inoltre potete vedere un pulsante dedicato alla riproduzione/pausa del video: in realtà è un'immagine che, se cliccata, richiama una funzione scritta nel linguaggio Javascript. Il file al quale la index fa riferimento per questo frame è video_frameRM.htm.

slideframe: qui c'è la visualizzazione delle slide e la possibilità di passare da una slide all'altra attraverso due pulsanti appositi. Per ogni immagine c'è la corrispondente pagina htm che rende visibile l'immagine stessa e i pulsanti appena descritti. Questi file htm come le immagini sono nominati con degli interi. E' importante per la fase successiva di conversione al sistema Lode capire che non necessariamente la pagina 1.htm richiama l'immagine .jpg omonima.

timeline_frame: è l'unico frame in cui è presente un applet Java. E' grazie a questo che si può ascoltare e riascoltare una parte di lezione semplicemente cliccando nel punto desiderato della timeline. Sono presenti dei marcatori temporali che identificano il passaggio da una slide all'altra. Passaggio che avviene in automatico: appena la lezione arriva ad un marcatore il frame slideframe visualizerà una nuova immagine. Il file che gestisce queste funzionalità è timeline_frameRM.htm. In esso sono memorizzati tutti i dati necessari al corretto funzionamento del sistema: nome delle singole slide e tempo in cui deve essere visualizzata.

link_frame: qui siamo di fronte ad un menù vero e proprio che ci mostra tutti i nomi delle single slide con la possibilità di portare il filmato e immagine con un semplice click al punto desiderato. In questo frame sono nuovamente memorizzati tutti i nomi delle slide e per ognuna è previsto l'evento onclick. Questo richiamerà una funzione che porterà il video ai secondi passati come parametro. Questo farà in modo che sulla timeline si arrivi ad un marcatore e che anche l'immagine visualizzata in slideframe venga cambiata.

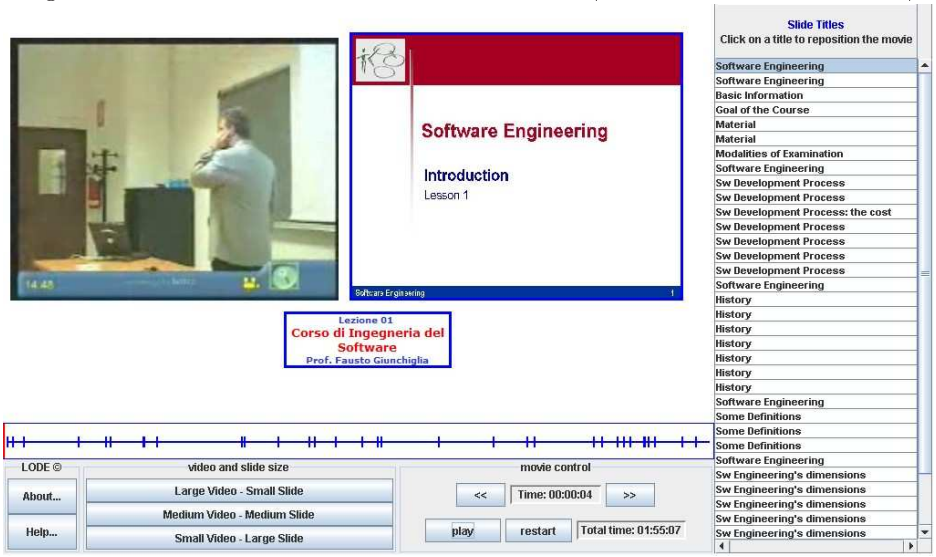
1.3 Il sistema Lode (versione Javascript+Java)

Questa versione di Lode permette agli utenti di poter visualizzare correttamente una lezione soltanto se sul proprio computer sono stati installati i plug-in JAVA (JDK raccomandata la 1.5 o successiva, anche se funziona anche con la 1.4) e QuickTime. Il browser di riferimento è Firefox che deve avere Javascript abilitato. Purtroppo non c'è la ossibilità di utilizzare il sistema in ambito linux: non vi è infatti tuttora un plug-in per QuickTime disponibile. Questo è sicuramente uno svantaggio notevole per i tempi che corrono: ormai sono molte le persone che si dirigono verso l'open source.

La versione corrente prevede una pagina introduttiva: LEGGIMI.html dalla quale è possibile accedere alla lezione vera e propria oppure fare un test per vedere se il proprio browser soddisfa i requisiti.

La pagina principale è chiamata index.html. Questa è il cuore del sistema e si presenta come in Figura 2:

Figura 2: La pagina principale del sistema Lode (versione Javascript + Java)



La pagina si presenta in visualizzazioni multiple. In particolare vi sono 5 parti ben distinte:

1. Dedicato al video da riprodurre. Nel dettaglio si tratta di un <object> in html, il browser lo riconosce come una pagina incorporata . Questo oggetto visualizza il video che, essendo in formato mp4, necessita del plug-in QuickTime per poter essere visto correttamente.

2. Dedicato alle informazioni sul corso e sulla lezione. In particolare vengono riportati titolo del corso, numero della lezione e il professore che la tiene. In questo caso siamo di fronte ad un vero e proprio frame. Nel codice troveremo <iframe>. Questo infatti

segna come sorgente il file title.html. E' qui che ci sono le informazioni scritte in precedenza.

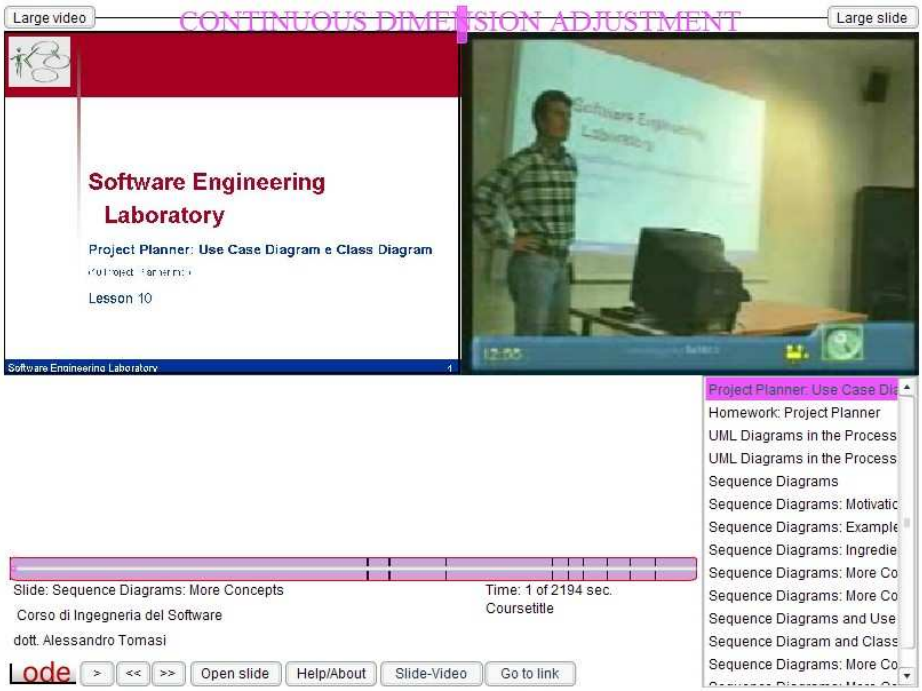
3. Dedicato alla timeline attraverso la quale è possibile muoversi alla posizione desiderata della lezione. E' un'applet Java (programma che viene eseguito grazie alla java virtual machine). Da qui la necessità del plug-in Java appunto. Come per ePresence l'applet gestisce la possibilità di cliccare in un punto qualsiasi della timeline per poter giungere all'istante della lezione desiderato. Anhe in questo caso ci sono dei marcatori temporali corrispondenti alle singole slide. Inoltre ci sono altri pulsanti che permettono di far fermare e ripartire la riproduzione del video oppure di saltare ai 15" precedenti o successivi dal punto in cui si è della lezione in quel momento. Nell'applet vengono contenute tutte le informazioni sulle singole slide e sui secondi corrispondenti del video in cui devono essere visualizzate. Infine è grazie a questo applet che si possono modificare le dimensioni del video e delle slide attraverso pulsanti previsti proprio per questa funzionalità.

4. Dedicato ai sottotitoli della lezione che spesso corrispondono ai titoli delle slide. Anche in questo caso siamo di fronte ad un'applet Java grazie al quale, cliccando su un qualsiasi sottotitolo si può arrivare al punto in cui esso viene esposto dal professore e la slide nella parte 3 verrà cambiata.

5. Dedicato alle slide del corso. Quest'ultima parte non fa altro che visualizzare l'immagine corrispondente all'istante in cui ci si trova della lezione. Le immagini sono tutte presenti in una sottocartella del sistema ed hanno come nome degli interi spesso progressivi. A differenza di ePresence in questo caso non ci sono pagine htm che visualizzano le slide: i file jpg vengono richiamati direttamente dall'index.

1.4 Il sistema Lode (versione Flash)

Figura 3: La pagina index.html del sistema Lode (versione Flash)



E' possibile fruire di questa versione solamente se si ha a disposizione un browser con correttamente installato il plug-in flash.

In questo caso non è presente una pagina introduttiva come nella precedente versione in quanto non c'è nessuna difficoltà per configurare il browser. Difficoltà che magari con la versione precedente potevano presentarsi.

Alla lezione si arriva immediatamente grazie alla pagina index.htm raffigurata in Figura 3.

Questa non presenta più tanti oggetti, frame o applet vari: la pagina è molto semplice, è formata da una ventina di righe di codice e da un solo oggetto che fa riferimento al file .swf (il programma scritto in flash). Questo è il file più importante che gestisce l'intera finestra del browser:

Le funzionalità sono praticamente le stesse della versione precedente:

- visualizzazione del video della lezione (questa volta in formato .flv) e della slide di cui si sta parlando (che sono tutte presenti in una sottocartella del sistema)

- visualizzazione di alcune informazioni sulla lezione: nome del corso, numero della lezione e professore che la tiene;
- lista dei sottotitoli fatta a menu con la possibilità di cliccare su uno di essi per portarsi al punto corrispondente del video
- timeline suddivisa da marcatori temporali con la possibilità di portarsi con un semplice click alla posizione desiderata
- pulsanti per il play/pause del video ed altri per farlo saltare al 10 secondi precedenti o successivi

Ci sono delle piccole migliorie come la possibilità di poter scegliere grazie ad un cursore la dimensione desiderata di slide e video o come il pulsante che ci permette di aprire e visualizzare l'immagine nella sua grandezza naturale o ancora la possibilità di avere video a destra e slide a sinistra o viceversa.

La struttura è quindi definita nel file in flash. Questo però si affida ad altri file per poter estrapolare i dati necessari ad ogni lezione. Il file principale è il data.xml che contiene tutte le informazioni necessarie per ogni singola slide...

```
<slide>
<tempo>0</tempo>
<titolo>Project Planner: Use Case Diagram</titolo>
<immagine>img/1.jpg</immagine>
</slide>
```

...ma anche informazioni riguardanti il video...

```
<video>
<nome>video.flv</nome>
<starttime>0</starttime>
</video>
```

... corso e professore che ha tenuto la lezione

```
<info>
<corso>Course <br/>Title</corso>
<titolo> Corso di Ingegneria del Software</titolo>
<professore>dott. Alessandro Tomasi</professore>
<dinamic_url>http://www.urldianmico.it/ciao</dinamic_url>
</info>
```

Vi sono quindi molti meno file rispetto al sistema standard descritto in precedenza.

E' indubbiamente un sistema più dinamico, più leggero e veloce. Ll'abbandono di Java ha ridotto in maniera considerevole anche i tempi di caricamento e visualizzazione di una lezione che ora appare subito disponibile nel browser senza attendere il caricamento della Java virtual machines.

L'altro aspetto da non sottovalutare è la portabilità del sitema: non c'è più infatti l'impossibilità di vedere una lezione su un sistema linux e questo è un vantaggio notevole che da più valore al sistema.

2 Conversione fra sistemi di digitalizzazione

2.1 Da ePresence a Lode (versione Js+J)

2.1.1 Introduzione

Qui inizia il mio lavoro. Come detto ePresence è un sistema già esistente da tempo. Lode è più recente e si richiede la necessità di recuperare tutte le lezioni presenti in ePresence e convertirle in Lode (js+J).

In particolare ecco i corsi che si devono convertire:

- Architettura degli elaboratori (17 lezioni)
- Laboratorio di Algoritmi e Strutture Dati (12 lezioni)
- Laboratorio di Informatica: Programmazione 2 (7 lezioni)
- Laboratorio di Sistemi Operativi (7 lezioni)
- Programmazione 2 (23 lezioni)
- Sistemi Distribuiti (4 lezioni)
- Ingegneria del Software (18 lezioni)

Per prima cosa ho prelevato tutte le lezioni in ePresence dal server Ortles presente in facoltà ed ho copiato tutto sul mio computer in modo da poter lavorare più autonomamente. Parecchio tempo è stato speso per questa operazione ma a fine giornata tutti i 15Gb di lezioni erano memorizzati nel mio hard disk.

Ho diviso l'operazione di conversione in due lavori ben distinti:

1. conversione video
2. conversione strutturale

La prima richiede parecchio tempo di elaborazione mentre la seconda necessita di tempo per la progettazione e costruzione di uno script. Ho quindi pensato di far lavorare fin da subito il computer lasciandolo spesso acceso durante la notte per effettuare la conversione dei video. Nel frattempopensavo allo script per la seconda conversione. La scelta è stata azzeccata: in questo modo ritengo di aver ottimizzato i tempi in modo ideale.

Al termine delle due conversioni ho spostato i video nelle cartelle previste da Lode ed il lavoro poteva ritenersi così concluso.

2.1.2 La conversione video

In questo ambito mi sono affidato a SUPER: un programma free funzionante sotto Windows che mi garantiva una buona qualità video ma soprattutto la conversione diretta dal formato .rm a .mp4 senza nessun problema. Il programma richiede soltanto i file in input ed il formato dei file in output. Inoltre mi veniva richiesto il formato audio e c'era la possibilità di cambiare varie impostazioni sul video e l'audio dei file in output.

Devo ammettere che per ogni video il tempo di conversione si è rivelato molto lungo ma alla fine ho ottenuto tutti i file nel formato desiderato (mp4 appunto).

2.1.3 La conversione strutturale

Inizialmente è stata individuato la struttura di base del sistema Lode con cartelle, pagine e file comuni a tutte le lezioni. Il risultato era un sistema Lode incompleto in cui mancavano tutte le immagini ed il file video. Inoltre la pagina index non era completa: erano presenti solamente le linee di codice HTML comuni a tutte le lezioni ma mancavano tutti i dati relativi ad ogni slide. Incompleto anche il file title.html in attesa di tutti i dati rigurdanti il corso, la lezione e professore che la tiene.

Successivamente ho costruito lo script vero e proprio di conversione. Questa volta ho lavorato in un sistema linux creando uno script bash: un insieme di comandi Unix finalizzato a svolgere un determinato compito.

Lo script inizialemente copia le immagini delle slide necessarie nella cartella prevista. Qui bisogna portare particolare attenzione: come descritto in precedenza infatti i due sistemi differiscono in questo punto. Lo script deve infatti aprire ogni singolo file htm di ePresence e salvarsi il nome dell'immagine che visualizza nel frame. Il nome salvato corrisponde all'immagine che doobbiamo copiare rinominandola però con il nome della pagina htm. In generale le pagine htm fanno riferimento all'immagine con lo stesso nome ma qualche volta ciò non accade. Ecco le linee di codice che gestiscono il problema:

```
//vado nella cartella delle slide e delle pagine htm
cd $NomeCartella/Data/Slides/
//salvo i nomi di tutte le pagine htm in un file
ls *.htm* > corrispHtml.txt
//conto quante pagine ci sono
NR=$(wc -l 'corrispHtml.txt'| cut -f1 -d" ")
//per ogni pagina
for ((d=1;d<=$NR;d++));do
    M=$(head -$d corrispHtml.txt| tail -1)
    /* prendo la riga in cui è presente il riferimento
       all'immagine*/
    G=$(less $M | grep .jpg)
    //ricavo il nome dell'immagine
    A=$(echo `expr match "$G" ".*window.open('\([0-9]*\).jpg"`)
    //ricavo il nome della pagina htm
    B=$(echo `expr match "$M" "\([0-9]*\).htm"`)
    //copio l'immagine dandole il nome della pagina htm
    cp $A.jpg ../../../$NomeCartella-lode/content/data/
              content_frame/$B.jpg
done
```

Le immagini sono state riportate correttamente. Ora dobbiamo estrapolare tutti i dati relativi alle singole slide di ePresence ed andare a completare la pagina index di Lode.

In particolare mi serve il file timeline_frameRM.htm : grazie a questo file riesco a risalire a tutti i dati relativi alla singola slide: durata in secondi della lezione, numero totale di slide, titolo, numero e secondi in cui viene richiamata ogni slide. La parte di codice html che ci interessa è la seguente:

```
<param name="totalSec" value="6907"></param>
...
<param name="totalSlides" value="37"></param>
...
#ripetuta per ogni slide
<param name="slide19" value="3631"></param>
...
#ripetuta per ogni slide
<param name="slideName6" value="Modalities"></param>
```

Con questo codice nello script riusciamo ad estrapolare i dati che vogliamo:

```
//memorizziamo la durata in secondi del video
C=totalSec
A=$(grep $C $file)
SEC=$(echo `expr match "$A" '.*'$C' *" *value *= *
      " *\([0-9]*\) *"'`)
//salviamo il numero di slide totale
C=totalSlides
A=$(grep $C $file)
SLIDE=$(echo `expr match "$A" '.*'$C' *" *value *= *
        " *\([0-9]*\) *"'`)
//qui salvo in un vettore il titolo delle varie slide
for ((d=0;d<$SLIDE;d++));do
    F=slideName$d
    G=$(grep $F $file)
    VETT_SL_NAME[$d]=$(echo `expr match "$G" ".*\$F *\" value
                      =\"\([a-zA-Z~0-9'\.:?&;,\*\/\(\)\!#=+-]
                      *\)\""` | sed s/~/\ /g)
done
//con lo stesso metodo in questa parte ricavo il tempo in cui
//ogni slide deve essere visualizzata
for ((d=0;d<$SLIDE;d++));do
    F=slide$d
    G=$(grep $F $file)
    VETT_SL_NUMB[$d]=$(echo `expr match "$G" '.*'$F' *" *
                            value *= *" *\([0-9]*\) *"'`)
done
```

A questo punto ci manca da ricostruire il file title.html e per farlo ci sevono i dati relativi al nome del corso, numero della lezione e professore che la tiene. Il file di ePresence che ci interessa video_frameRM.htm. In esso le linee di codice utili sono le seguenti:

```
<font face="Verdana" size="2" color="#000080">
```

```
<b>Lezione 01 - Corso di Ingegneria del Software</b></font>
...
<font face="Verdana" color="#000080" size="1">
<b>Prof. Fausto Giunchiglia</b></font>
```

Lo script va ad estrarre i dati nello stesso modo descritto in precedenza:

```
H=000080
I=$(grep $H $file2)
NLEZIONE=$(echo ‘expr match "$I" ’.*’$H’ *" *> *< *b *>\
                 ([[:alnum:][:blank:]\-\(\)\]*\) *\-’‘)
CORSO=$(echo ‘expr match "$I" ’.*’$H’ *" *> *< *b *>[[:alnum:]
          [:blank:]\-\(\)\]*\-\([[:alnum:][:blank:]:\-]*\)<’‘)
M=$(echo ${I##*00008})
PROF=$(echo ‘expr match "$M" ’0 *" size="1"><b> *\([[:alnum:]
                         [:blank:]\-\.]*\)<’‘)
```

Come si può notare il comando più utilizzato nello script è expr che, con gli opportuni parametri ed espressioni regolari, mi ha permesso di fare il parsing delle stringhe che desideravo ed estrarre le informazioni necessarie.

Ora abbiamo tutto ciò che ci serve e dobbiamo completare i file index.htm e title.htm di Lode.

Per farlo mi sono fatto aiutare dai seguenti comandi:

m4: mi ha permesso di inserire il contenuto di un file di testo in un altro file di testo semplicemente inserendo in un punto specifico il nome del file da includere scrivendo include(filename);

sed: l’ho utilizzato per sostituisce una stringa con un altra nei file che mi interessavano.

In pratica ecco come si presentava il file index.html:

```
...
<param name="totSec" value="#BBBBBB#">
<param name="tickCount" value="#CCCCCC#">
...
<!-- ticks definition -->
include(tickDefinition.txt)
...
```

Rimane da costruire il file tickDefinition.txt ma ho tutto quello che mi serve. Un semplice ciclo for scriverà una riga per ogni slide (come il sistema Lode vuole) con tutte le informazioni necessarie:

```
for ((d=0;d<$B;d++));do
    echo ’<param name="tick’$d’" value="0;0;0;’$d’;’
          ${VETT_SL_NUMB[$d]}’;’${VETT_SL_NAME[$d]}’;
          core.Slide;">’ >> tickDefinition.txt
```

```
done
```

Ora eseguiamo i comandi descritti in precedenza e ricostruiamo il file index correttamente:

```
m4 indexNeutro.html > indexApp.html
sed s/\#AAAAAA\#/$movieName/g indexApp.html > indexApp2.html
sed s/\#BBBBBB\#/$SEC/g indexApp2.html > indexApp3.html
sed s/\#CCCCCC\#/$SLIDE/g indexApp3.html > index.html
```

Per il file title.html la procedura è la stessa. Ecco il file incompleto:

```
<DIV class="title">
<SPAN class="title line1">#AAAAAA#</SPAN><BR>
<SPAN class="title line2">#BBBBBB#</SPAN><BR>
<SPAN class="title line3">#CCCCCC#</SPAN>
</DIV>
```

Con tre sed andiamo ad inserire i dati necessari:

```
sed s/\#AAAAAA\#/$NLEZIONEg titleNeutro.html > titleApp.html
sed s/\#BBBBBB\#/$CORSO/g titleApp.html > titleApp2.html
sed s/\#CCCCCC\#/$PROF/g titleApp2.html > title.html
```

Ora abbiamo la lezione convertita nel formato Lode standard.

L'ultima operazione sarà quella di inserire il video nel path richiesto dalla struttura di Lode. Le lezioni sono molte ed anche un semplice copia-incolla può diventare noioso e dispendioso in termini di tempo se ripetuto parecchie volte. Per questo ho utilizzato un terminale in linux per effettuare questa operazione e assicuro che i tempi si sono ridotti di molto.

2.1.4 Difficoltà riscontrate

Molti erano i programmi che potevano fare la conversione video da RealMedia a QuickTime direttamente ma purtroppo a pagamento. Ho pensato che non valesse la pena acquistare software per un'operazione che si faceva solo per questo scopo. Trovare un programma free che potesse soddisfare le mie esigenze è stato piuttosto difficile ma alla fine posso ritenermi soddisfatto del software utilizzato.

Lo script descritto in precedenza è stato quello più usato per la conversione di tutti i corsi. Non tutte le lezioni in ePresence però presentavano la stessa struttura. Questo mi ha fatto rivedere lo script più volte facendo delle piccole modifiche ed adattandolo ai singoli corsi che differivano dallo script descritto.

2.2 Da Lode (versione Js+J) a Lode (Flash)

Inquesto momento tutte le lezioni sono nel formato Lode standard. Ora però è necessario fare un'ulteriore conversione al più recente sistema progettato in Flash. Questa operazione è stata molto più veloce della precedente.

Conversione video e strutturale questa volta le ho inglobate in un unico script. Ho infatti notato che i video vengono trasformati molto più velocemente da mp4 a flv grazie a ffmpeg rispetto alla precedente conversione da rm a mp4 con Super:

ffmpeg -i movie.mp4 video.flv

Prima di questa linea di codice, come fatto nella conversione precedente, mi sono costruito una struttura base del sistema in Flash: l'insieme di files e cartelle che devono necessariamente esserci per tutte le lezioni.

Lo script va avanti copiando le immagini delle slide presenti nella lezione in Lode Standard nella cartella designata di Lode Flash. Inoltre vengono copiati e salvati da una parte i file title.html e index.html di Lode Standard che ci serviranno per ottenere tutte le informazioni necessarie per passare alla versione flash.

Come già detto in precedenza il file che contiene tutte le informazioni in Lode Flash è il data.xml. Sarà quindi questo il file che dovremmo costruire una volta ricavati i dati dai due file di Lode standard.

La struttura di Lode ci facilita le cose rispetto a ePresence. Infatti ora siamo certi che tutte le lezioni hanno la stessa struttura e che quindi basta creare uno script e quello andrà bene per tutte.Anche trovare le varie informazioni è molto più semplice... il file index.html è strutturato in modo tale che su ogni riga ci siano tutto ciò che vogliamo su ogni singola slide:

```
..
<param name="tick3"
 value="6;2006;3;3;195;Goal of the Course;core.Slide;">
..
```

In ordine in grassetto divisi dal ";": numero slide, secondi in cui compare, titolo della slide.

I dati su ogni singola slide sono sempre quelli presenti nel file index.html (in questo caso il codice è lo stesso di quello utilizzato per la crazione delle query di inserimento delle singole slide visto in precedenza). Estrapoliamo i dati che ci servono con il seguente script:

```
for ((d=0;d<$NSLIDE;d++));do
    F=tick$d\"
    G=$(grep $F $file)
    VETT_SL_FILE_NAME[$d]=$(echo `expr match "$G" ".*\$F *
           value=\"[0-9]*\;[0-9]*\;[0-9]*\;\([0-9]*\)\;"`)
    oioi2=${VETT_SL_FILE_NAME[$d]}
    VETT_SL_NUMB[$d]=$(echo `expr match "$G" ".*\$F *value=
```

```
        \"[0-9]*\;[0-9]*\;[0-9]*\;\$oioi2\;\([0-9]*\)\;"‘)
    VETT_SL_NUMB[$d]=$(echo ‘expr match "$G" ".*\$F.*\;\(
                            [0-9]*\)\;"‘)
    oioi=${VETT_SL_NUMB[$d]}
    VETT_SL_NAME[$d]=$(echo ‘expr match "$G" ".*\$F.*\;\
              $d\;\$oioi\;\([a-zA-Z~0-9’\.\ :?&,;\*\/\(\)
              \!#=+-]*\)\;core"‘ | sed s/~/\ /g)
done
```

E’ molto simile allo script che fa la conversione da ePresence a Lode. Questa parte di codice salva in 3 vettori i dati relativi ad ogni slide presente nel file index.html: nome del file della slide, titolo dela slide e tempo del video in corrispondenza del quale la slide deve essere visualizzata.

Corso, lezione e professore vengono estrapolati dal file title.html sempre con il comando “expr”:

```
file2=title.html
H=’title line’
I=$(grep $H"1" $file2)
L=$(grep $H"2" $file2)
M=$(grep $H"3" $file2)
Z=’title line1">’
LEZ=$(echo ‘expr match "$I" ’.*title line1">\([[:alnum:]
                        [:blank:]\-\(\)\.\]*\)<’‘)
CORSO=$(echo ‘expr match "$L" ’.*title line2">\([[:alnum:]
                          [:blank:]\-\(\)\.\]*\)<’‘)
PROF=$(echo ‘expr match "$M" ’.*title line3">\([[:alnum:]
                         [:blank:]\-\(\)\.\]*\)<’‘)
```

A questo punto abbiamo tutti i dati che ci servono... ricostruiamo quindi il file xml:

```
echo "<lezione>" >> data.xml
for ((d=0;d<$B;d++));do
P=$[${VETT_SL_FILE_NAME[$d]}+$COUNTSHIFT]
I=$[$B-1]
echo "" >> data.xml
echo "<slide>" >> data.xml
echo "<tempo>"${VETT_SL_NUMB[$d]}"</tempo>" >> data.xml
echo "<titolo>"${VETT_SL_NAME[$d]}"</titolo>" >> data.xml
echo "<immagine>img/"$P".jpg</immagine>" >> data.xml
echo "</slide>" >> data.xml
echo "" >> data.xml
done
echo "" >> data.xml
echo "<video>" >> data.xml
```

```
echo "<nome>video.flv</nome>" >> data.xml
echo "<starttime>0</starttime>" >> data.xml
echo "</video>" >> data.xml
echo "" >> data.xml
echo "<info>" >> data.xml
echo "<corso>Course <br/>Title</corso>" >> data.xml
echo "<titolo>"$CORSO"</titolo>" >> data.xml
echo "<professore>"$PROF"</professore>" >> data.xml
echo "<dinamic_url>#</dinamic_url>" >> data.xml
echo "</info>" >> data.xml
echo "</lezione>" >> data.xml
```

Infine copiamo i files data.xml ed il video convertito in precedenza nelle cartelle designate dal sistema Flash. Abbiamo ora la lezione nel formato più recente di Lode.

3 Definizione di un database per l'archiviazione del materiale

3.1 Introduzione

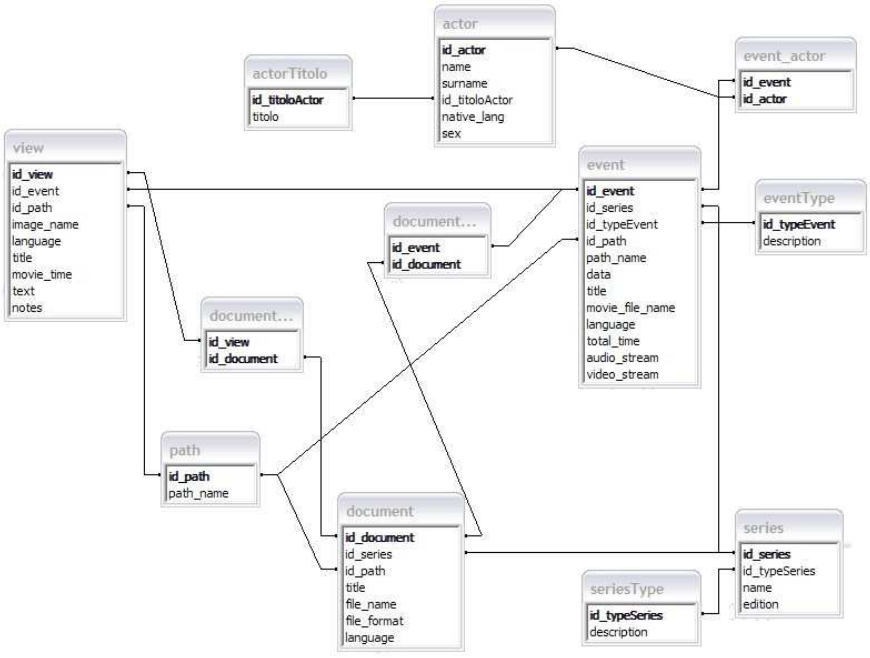
Ora che tutte le lezioni sono in Lode è necessario creare una base di dati che possa contenere tutte le informazioni riguardanti ogni corso,lezione,slide presenti in modo tale che sia possibile trovare uno specifico argomento o lezione con semplicità ed efficienza. Il database in un primo momento verrà usato unicamente dal responsabile del progetto Lode in locale ma nulla vieta che in futuro (con qualche accorgimento) possa essere disponibile online per fare in modo che chiunque possa visualizzare le tante informazioni in esso contenute.

Il database è MySQL: molto facile da utilizzare e imparare ma anche molto veloce in termini di tempi di risposta alle varie query.

3.2 Struttura del database

Lo schera relazionale del database è in Figura 4

Figura 4: Schema relazionale del database



Ecco le varie tabelle:

path: questa tabella permette di tenere traccia di vari path che poi verranno usati in altre tabelle. In realtà il suo utilizzo torna utile nel caso in cui il sistema dovesse cambiare la propria struttura: sarà solamente necessario modificare i dati in questa tabella. Nel nostro caso abbiamo preso il sistema Lode standard come riferimento.

eventType: qui vengono inseriti i vari tipi di eventi (es. lezioni)

seriesType: tabella dedicata ai tipi di corsi (corso universitario, seminario,...)

actorTitolo: qui si trovano le qualifiche delle varie persone che tengono le lezioni.

actor: vengono memorizzati tutti i dati riguardanti i professori che tengono le lezioni

Ci sono linoltre tabelle con delle chiavi esterne. Queste tabelle presentano un'associazione uno a molti con le tabelle referenziate (cioè un elemento della tabella può essere associata ad uno solo della tabella referenziata e questo può essere associato a più elementi della tabella che la referenzia)

series: vengono salvati i dati relativi ai vari corsi, seminari, in essa vi è una chiave esterna con referenziata la tabella seriesType

document: eventuali documenti, materiale vario compaiono in questa tabella. In essa ci sono 2 chiavi esterne che referenziano le tabelle series e path. Ciò vuol dire che un elemento della tabella document può essere associato ad una sola serie e path). In effetti un documento può appartenere solo ad un corso e può essere memorizzato solamente in un path ben preciso

event: è una delle tabella più corpose. Memorizza le varie lezioni. Qui si trovano i dati relativi al video (path, tempo totale, lingua, audio e video stream). In questo caso ci sono ben 3 chiavi esterne relative alle tabelle series, eventType e path. Un elemento di event può dunque essere associato ad una sola serie, eventType e path. Infatti una lezione può essere solo di un tipo e può avere solo una determinata struttura

view: E' la tabella più corposa in assoluto... ogni slide infatti occupa una tupla della tabella memorizzando il proprio titolo, il nome del file, il tempo in cui deve comparire ed altro ancora. In questo caso ci sono due chiavi esterne che referenziano event e path. Ad esempio una slide può essere associata ad una sola lezione e path.

Infine ci sono le tre tabelle che permettono di rappresentare correttamente l'associazione binaria molti a molti fra due tabelle descritte in precedenza (ogni elemento di una tabella può essere associato a più elementi dell'altra tabella e viceversa).

document_view: i suoi attributi sono solamente l'id di document e l'id di view che assieme fanno da chiave primaria. Ci saranno due chiavi esterne che referenziano le tabelle document e view. Grazie a questa tabella un elemento di document può essere associato a più elementi di view e viceversa. In tal modo più documenti possono essere associati ad una slide e viceversa.

document_event: i suoi attributi sono solamente l'id di document e l'id di actor che assieme fanno da chiave primaria. Ci saranno due chiavi esterne che referenziano le tabelle document e event. Grazie a questa tabella un elemento di document può essere associato a più elementi di event e viceversa. Solo così vari docuemnti possono essere attribuiti ad una lezione e viceversa.

event_actor: i suoi attributi sono solamente l'id di event e l'id di actor che assieme fanno da chiave primaria. Ci saranno due chiavi esterne che referenziano le tabelle actor e event. Grazie a questa tabella un elemento di actor può essere associato a più elementi di event e viceversa. Pensate ad esempio ad una lezione in cui sono presenti due professori... senza questa tabella ciò non sarebbe memorizzabile nel database.

3.3 Inserimento dei dati

3.3.1 Introduzione

Il database è pronto, dobbiamo solamente estrapolare i dati di tutte le lezioni in Lode.

Data la struttura del db ci risulta semplice inserire series, seriesType ed eventType per non parlare di path, actorTitolo e actors:

```
INSERT INTO ‘eventType‘ VALUES (1, ’Lecture’);
INSERT INTO ‘seriesType‘ VALUES (1, ’University Course’);
INSERT INTO ‘series‘
VALUES ("LABPROG2_", 1, "Lab. di Inf.: Programmaz. 2",NULL);
...
INSERT INTO ‘path‘ VALUES (1, ’/content/data/content_frame/’);
INSERT INTO ‘path‘ VALUES (2, ’/content/data/’);
INSERT INTO ‘actorTitolo‘
VALUES (’UNITNProf’, ’Professore Università di Trento’);
INSERT INTO ‘actor‘
VALUES (’maumar’,’Maurizio’,’Marchese’,’UNITNProf’,’ITA’,’M’);
...
```

Più laborioso e noioso risulta essere l'inserimento delle event per ogni singola series ma soprattutto delle view.

Anche in questo caso ci affidiamo a degli script in bashper estrapolare tutti i dati che ci servono.

3.3.2 script 1

Il primo molto semplice creerà una query di inserimento per tutte le lezioni di ogni corso come ad esempio:

```
INSERT INTO ‘event‘ (id_event,id_series,id_typeEvent,id_path,
                path_name,data,title,movie_file_name,language,
                total_time,audio_stream,video_stream)
```

```
VALUES
("ARCELAB1","ARCELAB", 1, 2,"Lezione01-lode",NULL,"Lezione 1",
 "movie.mp4",NULL,NULL, NULL, NULL);
```

Basta un ciclo che, per ogni corso, crea questa linea per il numero di lezioni in esso presenti..

3.3.3 script 2

Il secondo passa in rassegna il file index.html di lode e preleva da esso tutti i dati necessari.

Lo script è identico a quello visto nel paragrafo 2.2 usato per la conversione tra sistemi Lode. Questo memorizza i dati in tre vettori.

Ora basta creare la query di inseriemnto vera e propria:

```
echo 'INSERT INTO `view` (id_event,id_document,id_path,
            image_name,language,title,movie_time,text,notes)
      VALUES' >> tickDefinition.txt
for ((d=0;d<$B;d++));do
    P=$[${VETT_SL_FILE_NAME[$d]}+$COUNTSHIFT]
    I=$[$B-1]
    if [ "$d" = "$I" ];
        then

            echo '("'$IDLezione'", NULL, 1,"'$P'.jpg",NULL,
                 "'${VETT_SL_NAME[$d]}'",'${VETT_SL_NUMB[$d]}'
                 ,NULL, NULL);' >> tickDefinition.txt
        else

            echo '("'$IDLezione'", NULL, 1,"'$P'.jpg",NULL,
                 "'${VETT_SL_NAME[$d]}'",'${VETT_SL_NUMB[$d]}'
                 ,NULL, NULL),' >> tickDefinition.txt
    fi
done
```

Una piccola riflessione va fatta sulla terza riga ed in particolare sulla variabile $COUNTSHIFT. Nel file index nell'applet timeline c'è questa riga:

```
...
<param name="slideCountShift" value="1">
...
```

Ebbene questo valore serve a Java per capire quale immagine andare a visualizzare nel browser. Ad esempio se il valore di slideCountShift è 1 e il video è arrivato al punto in cui deve visualizzare la slide che nell'index è identificata dal numero 3, allora Java andrà a prendere il file 4.jpg. Se slideCountShift è 0 allora andrà a visualizzare 3.jpg. Di questo

dobbiamo tenere conto nello script ed è proprio la riga 3 che se ne occupa. La variabile COUNTSHIFT è stata già memorizzata in precedenza con le seguente righe nello script:

```
C=slideCountShift
A=$(grep $C $file)
COUNTSHIFT=
  $(echo `expr match "$A" '.*value *= *" *\([0-9]*\) *"'`)
```

Ecco quindi l'esempio di una query di inserimento:

```
INSERT INTO `view` (id_event,id_path,image_name,language,
                    title,movie_time,text,notes)
VALUES
("ARCELAB1", 1,"1.jpg",NULL,"Copyright",0,NULL,NULL),
("ARCELAB1", 1,"2.jpg",NULL,"Arch. degli Elab.",5,NULL, NULL);
```

4 Prototipo di annotazione dei video

4.1 Introduzione

Ricapitoliamo brevemente il tutto: siamo arrivati ad un punto in cui tutte le lezioni, corsi, seminari digitalizzate con i sistemi ePresence e Lode (formato standard) sono ora anche in formato Lode Flash. In prospettiva futura c'è la possibilità di pubblicare online tutte le lezione in quest'ultimo formato che abbiamo già descritto come più veloce ed efficiente.

E' in questo ambito che nasce l'idea di ampliare il sistema rendendo possibile la scrittura di commenti, opinioni o giudizi su una specifica lezione o slide. Vogliamo quindi che un qualsiasi utente da casa si scarica la lezione desiderata in formato flash presente online. A questo punto segue la lezione attraverso un comune browser e all'istante che ritiene opportuno clicca su un pulsante apposito che lo porta ad un sito internet dal quale può scrivere una qualsiasi frase riguardante l'istante scelto. E' evidente che ci dovrà essere una pagina di login e la possibilità di registrazione per l'utente. Una volta registrato, oltre che scrivere un commento c'è anche l'opportunità di una ricerca che dia la possibilità allo stesso di visualizzare anche le opinioni scritte da altri utenti.

Serve quindi:

- un database che contenga tutte le informazioni riguardanti gli utenti, le lezioni ed i commenti;
- le pagine web che andranno a formare il sito internet dal quale sarà possibile effettuare tutte le operazioni descritte in precedenza.

4.2 Il database

4.2.1 Introduzione

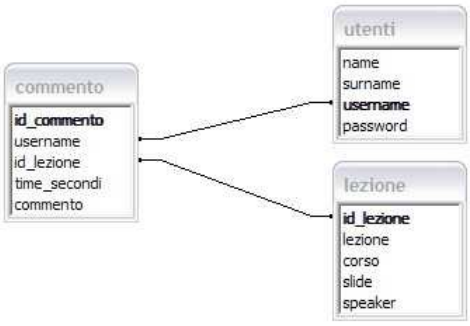
Questa è la parte fondamentale per la buona riuscita del progetto: a mio modo di vedere un'ottima struttura del database consente di risolvere molti problemi prevenendo parecchi errori e facilitando il lavoro successivo di inserimento e ricerca di elementi.

Anche in questo caso ho scelto MySQL come database per le stesse motivazioni espresse in precedenza

4.2.2 Struttura del database

La struttura è molto semplice: ci sono tre tabelle che contengono le informazioni necessarie. Lo schema referenziale è illustrato in Figura 5

Figura 5: Schema relazionale del database



UTENTE Questa tabella racchiude tutte le informazioni riguardanti ogni utente che intende registrarsi ed acceddere al sistema. Quindi nome, cognome, username e password. In questo caso non è necessario definire un id per la chave primaria che verrà identificata dall'attributo username per definizione univoco.

LEZIONE Qui viene memorizzato tutto ciò che ci interessa riguardo la lezione ma in particolare anche la slide, il corso ed il professore che la tiene. In questo caso è necessario un id che faccia da chiave primaria identificando unicamente ogni elemeneto.

COMMENTO L'opinione che un utente va a scrivere è contenuta in questa tabella nella quale dovrà esserci una chiave esterna che referenzia UTENTE ed un'altra che referenzia LEZIONE in modo tale da tener traccia di chi ha scritto il commento ed a quale lezione,slide,corso si riferisce. Qui ci deve essere anche un attributo riguardante i secondi della lezione referenziata in corrispondenza dei quali è stato scritto il commento. Infine è necessario anche qui un id che faccia da chiave primaria.

4.2.3 Osservazioni

In questo stadio embrionale del sistema la struttura risulta molto semplice. Nonostante ciò tutte le informazioni che interessano sono presenti.

Qualche dubbio può destare la tabella lezione che racchiude nome del professore, titolo della slide e titolo del corso. In questo modo se ci sono 10 slide per una lezione allora nella tabella ci saranno 10 elementi che differiscono solamente per il titolo della slide. Questo però diventerebbe un problema qualora un professore, ad esempio, avesse anche altri attributi: in tal modo questi dovrebbero essere ripetuti per ogni elemento di Lezione. Lo stesso vale per la lezione ed un corso: se oltre al nome del corso vorremmo memorizzare il fatto che si tratti di un seminario, ci si troverebbe davanti a delle duplicazioni inutili. Ogni elemento di lezione potrebbe essere così composto:

...

(id1,nome corso,corso UNITN, lezione,professore,sesso professore,nome slide1)

(id2,nome corso,corso UNITN, lezione,professore,sesso professore,nome slide2)

...

Così facendo ci si porta avanti corso UNITN e sesso professore per niente rendendo inefficiente il database.

In un futuro se lo si riterrà necessario (e quindi in aggiunta di attributi) basterà aggiungere una tabella Professore e una tabella Corso che verranno referenziate da Lezione con due chiavi esterne.

Un'altra osservazione può essere espressa riguardo l'id di Lezione. In previsione futura pensate a quanto potrebbe risultare utile trovare un collegamento tra questo database e quello costruito in precedenza che, ricordo, contiene tutte le informazioni che potremmo desiderare. Ebbene, ciò potrebbe essere fattibile proprio grazie all'id della lezione di questa base di dati che potrebbe rappresentare il riferimento che vogliamo per l'altro database.

4.3 Il sito internet

4.3.1 Le scelte

Abbiamo dunque un "raccoglitore" di tutti i dati che ci interessano. Ora è giunto il momento di costruire un'interfaccia web che possa permettere all'utente di usufruire, consultare questi dati. Serve dunque un linguaggio di script che ci permetta di collegarsi al database MySQL e credo che php ci possa aiutare in questo.

E' poi di fondamentale importanza dare un aspetto gradevole al sito internet dando la possibilità in futuro di apportare aggiornamenti senza dover cambiare ogni singola pagina presente. In questo caso mi affido ai fogli di stile.

MySQL–Php:

Php è un linguaggio di scripting con licenza open source concepito per la creazione di pagine dinamiche mentre MySQL è un database relazionale con diversi strumenti di amministrazione. Uno dei programmi più popolari è phpMyAdmin: richiede un server web come Apache HTTP Server ed il supporto del linguaggio php. Si può inoltre utilizzare facilmente tramite un qualsiasi browser.

Indubbiamente questa accoppiata è, al giorno d'oggi, una delle più diffuse in rete in quanto abbiamo a disposizione gratuitamente un linguaggio solido, capace di sopportare grandi carichi di lavoro, e un database dalle notevoli qualità tecniche.

Per poter testare il codice sul nostro computer di casa abbiamo bisogno di un webserver (io ho utilizzato Apache), del modulo Php e del server MySQL. Questi ci eviteranno il fastidio e il costo di dover trasferire le pagine su un server ogni volta che ne vorremo valutare la correttezza.

CSS

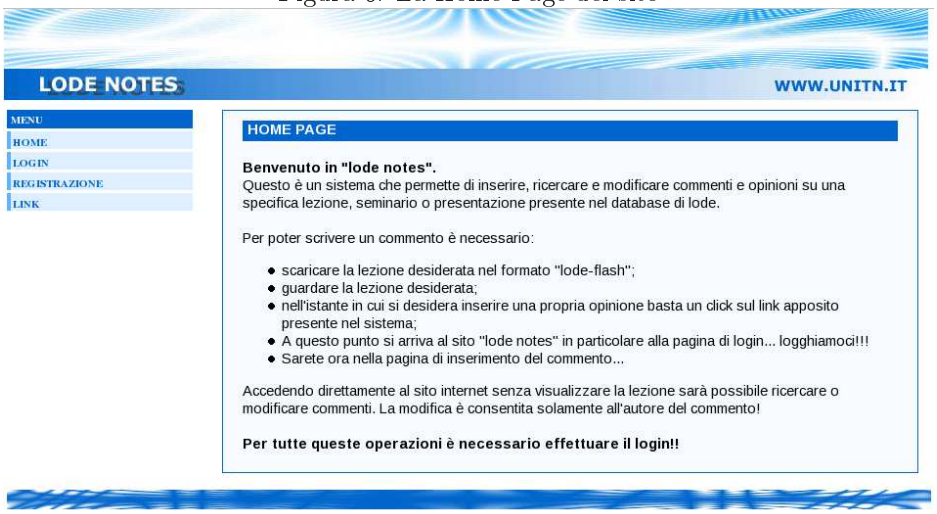
I fogli di stile a cascata (dall'inglese CSS Cascading Style Sheets), detti semplicemente fogli di stile, sono una tecnica che permette di fissare gli stili (per es. tipo di carattere,

colori e spaziature) da applicare ai documenti. Grazie a questi si possono separare i contenuti dalla formattazione e permettere una programmazione più chiara e facile da utilizzare, sia per me che per gli utenti.

C'è un grosso vantaggio che ho già accennato prima: ammettiamo che tutte le pagine facciano riferimento ad un foglio di stile che prevede lo sfondo di ogni pagina rosso. Se in futuro ci sarà la necessità di cambiare lo sfondo basta modificare solamente il file del foglio di stile senza nemmeno guardare il codice di ogni singola pagina. Il risultato sono pagine più leggere e facili da modificare, milioni di byte di banda risparmiati.

4.3.2 La struttura del sito

Figura 6: La Home Page del sito



L'impaginazione di una pagina web può avvenire in vari modi. Potevo ottenere un buon risultato usando le tabelle ma mi sono basato sui fogli di stile. Dalla diffusione dei fogli di stile la tendenza di creare layout senza l'uso di tabelle si è andata man mano consolidando, fino a diventare una realtà in continua espansione.

Un layout table-less si realizza attraverso l'uso dei div. La sua definizione è generic block-level element ossia contenitore generico block level. Il fatto che sia un elemento block-level ci garantisce il fatto che possa contenere qualsiasi tipo di elemento html. Inoltre, la sua presentazione naturale (quindi senza fogli di stile) è totalmente neutra: infatti questo elemento si presenta di default senza margini, bordi o padding. È quindi il contenitore per eccellenza per realizzare layout senza l'uso di tabelle: ad ogni div portante viene associata una sezione della pagina.

Ecco qui la struttura delle mie pagine:

```
<html lang="it">
<title>Lode notes</title>
```

```
<link rel="stylesheet" type="text/css"
  href="img_css/lodeNotes.css" title="default">
<body>
<div id="MainBoxInternal">
    <div id="Header">
    </div>
    <div id="MainPage">
        <div id="Bar">
            <div id="Menu">
            </div>
        </div>
    <div id="BoxPrincipale">
        <div class='TitoloBoxPrincipale'>
        </div>
    </div>
</div>
<div id="Credits">
</div>
</body>
</html>
```

Come potete notare nella riga 3 ho incluso il file .css. Questo non è altro che un foglio di stile esterno che definisce le varie regole che interessano al documento html. Nel nostro caso ecco l'esempio di com'è stato definito nel file .css il "Box Principale":

```
div#BoxPrincipale {
    background-color: #F4FAFF;
    border: 1px solid #0066CC;
    padding-left: 20px;
    padding-right: 20px;
    padding-top: 10px;
    padding-bottom: 20px;
    margin-left: 20px;
    background-attachment: fixed;
    background-repeat: no-repeat;
    background-position: right top;
    float: left;
    width: 670px;
    height: auto;
    margin-bottom: 15px;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 15px;
    margin-top: 10px;
```

```
}
```

Come potete vedere ci sono parecchi attributi presenti solamente nel file .css ma necessari a tutte le pagine del sito. Ovviamente per ogni div presente nella pagina html corrisponde una definizione nel file css. Il risultato finale della pagina base si può vedere in Figura 7:

Tutte le pagine sono quindi strutturate in questo modo. L'una dall'altra differiscono solamente per le parti identificate come "Box Principale" (nella quale verranno inseriti i vari form e i contenuti delle pagine) e come "Menu" (nella quale verranno visualizzati i link alle pagine che si possono visitare).

4.3.3 Gestiamo l'invio dei dati da parte dell'utente

Alcune pagine del nostro sito devono permettere l'invio di varie informazioni da parte dell'utente che gli permettono di fare ad esempio ricerche di un commento in base ad una stringa specifica. Questo avviene tramite l'elemento <FORM> che permette di definire un'area all'interno della quale un utente può inserire qualsiasi dato (numeri, testo, valori booleani).

```
<form name="" action="" method="">>
...
</form>
```

<FORM> ha vari attributi fra i quali:

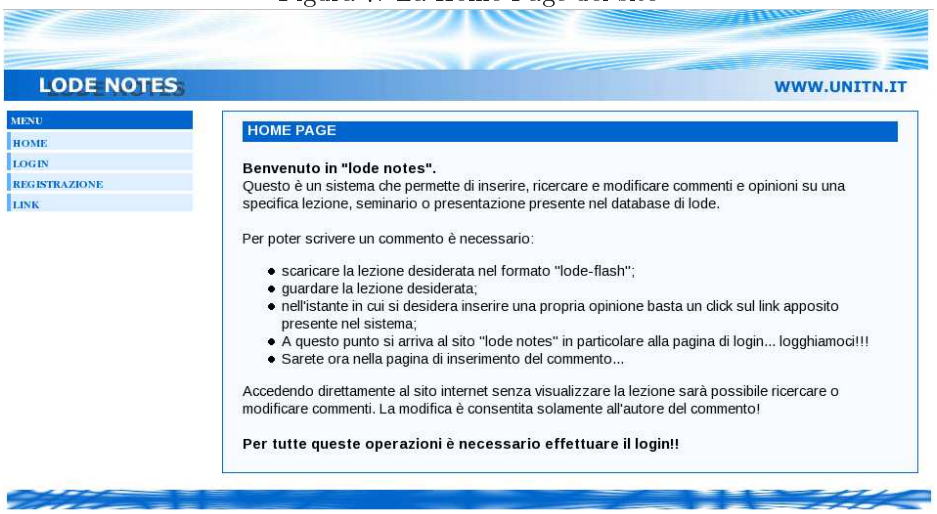
NAME:

Serve ad indicare il nome del form

ACTION:

E' l'indirizzo che individua lo script responsabile della ricezione e gestione dei

Figura 7: La Home Page del sito



dati trasmessi. Nel nostro caso sarà lo stesso script php che ha generato il form in precedenza. Ciò vuol dire che all'interno del file php ci dovranno essere dei controlli grazie ai quali si può capire se alla pagina si è giunti attraverso il form oppure da un altro link (es. dal menu)

METHOD:

Definisce la modalità in cui vengono trasmessi i dati. Può assumere valore GET o POST.

Nel primo caso la pagina di risposta viene contattata e i dati vengono inviati in un unico passo. Nell'URL della pagina di risposta possiamo quindi vedere tutti i dati inviati nella seguente forma:

```
funzioni.php?par1=prova&par2=prova2
```

In questo modo è possibile inviare dei dati anche senza ricorrere ad un form nella pagina html... vedremo in che modo questo possa accadere successivamente.

Nel metodo POST invece l'invio dei dati avviene in due passi distinti: prima viene contattata la pagina sul server che deve processare i dati, poi vengono inviati i dati stessi. Per questo motivo i parametri non compaiono nella URL (dunque se non si desidera che i parametri siano mostrati all'utente questo metodo è preferibile). Inoltre POST è il metodo più adatto ad inviare dati di una certa lunghezza, poiché con GET, per ragioni pratiche, non si possono inviare al server più di 256 byte di informazione.

Per quanto riguarda i campi del form il tag principale che permette di creare una forma di immissione dati è <INPUT>. Ogni <INPUT> dichiarato sarà un parametro da inviare al server. Gli attributi più importanti di <INPUT> sono:

NAME=nome Nome simbolico dei dati inviati al server. E' previsto per la maggior parte dei tipi di input, per far sì che i dati di ogni campo siano contraddistinti ognuno da un unico identificatore.

VALUE=valore Valore iniziale visualizzato all'interno del campo.

TYPE="checkbox/image/radio/password/text/submit/hidden/reset" Imposta il tipo di dati che il campo accetterà. per questo progetto ci interessano solamente:

- **password:** crea un campo di testo ove i caratteri immessi vengono visualizzati con asterischi (*). Ciò dà la possibilità di generare documenti ipertestuali interattivi in cui l'utente può digitare una password con la dovuta riservatezza.
- **Text:** crea un'area ipertestuale in cui l'utente può inserire una singola linea di testo.
- **Hidden:** crea un campo nascosto per l'utente. Il valore imposto ad un campo hidden può costituire l'informazione di "stato" per il server, cioè quella informazione che non deve essere modificata dal client.
- **Submit:** dà l'istruzione di invio dati. submit crea un pulsante ipertestuale, che può anche essere etichettato assegnando una stringa all'attributo VALUE. Premendo tale pulsante, l'utente invia al server tutti i dati che egli ha impostato nei campi.
- **Size:** Stabilisce in caratteri la lunghezza da visualizzare di un campo.

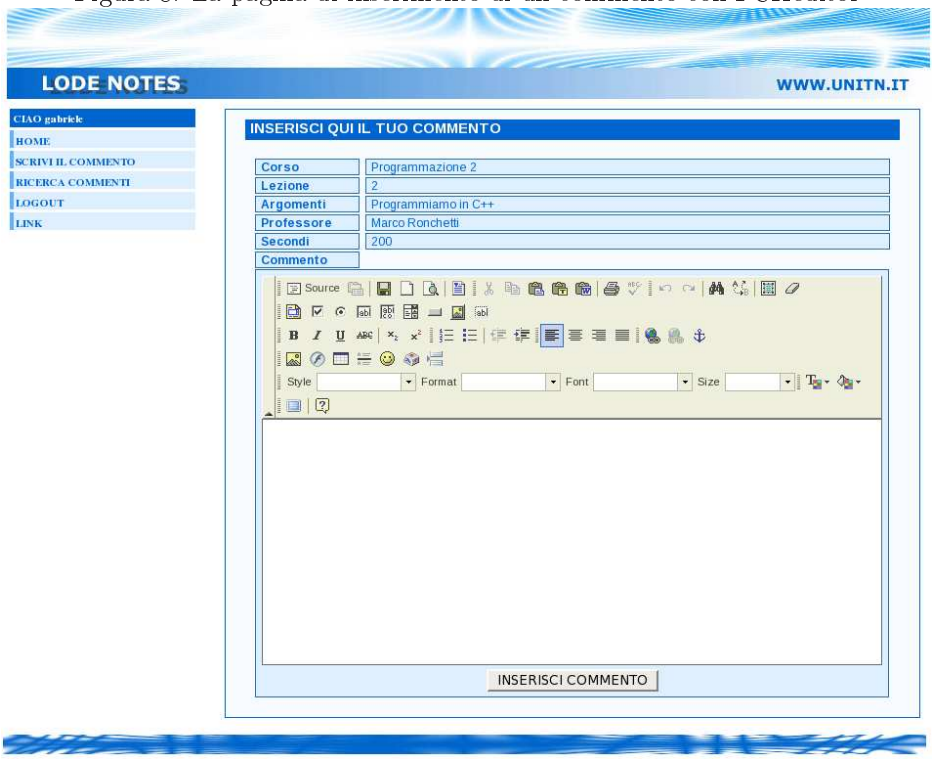
4.3.4 Un editor per scrivere i commenti: FCKeditor

l fulcro di tutto il sistema è la possibilità di scrivere commenti da parte di chiunque ne abbia il desiderio. Sarebbe molto riduttivo inserire una normale TEXTAREA nel form senza lasciare all'utente nessuna libertà sull' impaginazione, sugli stili, font del commento ma non solo. Per questo mi sono rivolto a FCKeditor: un editor HTML open source che consente di portare le principali funzionalità di un editor come Word sul web.

Ha molti pregi fra i quali la sua compatibilità con i più famosi browser web: Firefox, Mozilla e Netscape e funziona su Linux, Windows e Mac. Inoltre, a lato server, FCKeditor offre una completa integrazione con parecchi linguaggi di programmazione :ASP, ColdFusion, Java, Perl, Pyton, Lasso, Active-FoxPro e naturalmente PHP.

Si presenta all'utente come in Figura 8

Figura 8: La pagina di inserimento di un commento con FCKeditor



L'installazione è veramente semplice: basta scaricare l'ultima versione dal sito internet e scompattare il pacchetto in una qualsiasi directory interna al nostro web server.

Ora supponiamo che l'editor sia installato in /FCKeditor/ sul nostro sito web. La prima cosa da fare, nelle pagine a cui serve, è di includere il file che contiene i moduli di integrazione a PHP in cima alla pagina:

```
<?php
include("FCKeditor/fckeditor.php");
?>
```

Ora FCKeditor è pronto all'uso. Basta solamente inserire le seguenti righe di codice all'interno delle pagine interessate che creeranno un'istanza dell'editor (solitamente all'interno di un <FORM>):

```
<?php
$oFCKeditor = new FCKeditor('FCKeditor1');
$oFCKeditor->BasePath = '/fckeditor/';
$oFCKeditor->Value = 'Default text in editor';
$oFCKeditor->Create();
?>
```

In questo esempio "FCKeditor1" è il nome usato per identificare i dati scritti attraverso l'editor che verranno inviati attraverso il form.

4.3.5 Manteniamo lo stato: le sessioni

La sessione si può definire come l'arco di tempo in cui viene monitorata la connessione di un utente. Durante questo arco di tempo è possibile conservare informazioni sulla navigazione accessibili da ogni pagina collegata alla sessione. La sessione quindi inizia quando l'utente accede al sito e finisce quando lo abbandona, chiudendo il browser.

Per dare via a una sessione si utilizza la funzione

```
session_start()
```

Dal momento in cui verrà usata la sessione sarà attiva e potremo utilizzarla.

Per registrare variabili in essa basta inserirle nell'array associativo globale $_SESSION nel seguente modo:

```
$_SESSION['nome_var'] = "valore";
```

Da questo momento in poi l'array $_SESSION conterrà il valore "valore" associato all'indice "nome_var", il che può essere visto come l'aver salvato, all'interno della sessione, la variabile "nome_var" a cui è associato il valore "valore".

Una volta terminato il lavoro è opportuno poter distruggere una sessione. Useremo tre funzioni per fare questo:

- unset ($_SESSION['nome_var']); : il comando cancella il contenuto della variabile di sessione
- $_SESSION=array(); : viene cancellato il contenuto dell'array globale $_SESSION e quindi tutte le variabili di sessione;
- session_destroy(); :distrugge la sessione... elimina fisicamente una sessione, in modo da poterne creare una nuova in seguito

A ciò va aggiunto il fatto che le sessioni sono associate ad un identificativo univoco, che viene generato quando si da inizio alla sessione e che solitamente viene conservato in un cookie nella variabile PHPSESSID. Quando i cookie sono disabilitati dobbiamo provvedere noi stessi al mantenimento di questo valore, a meno che in PHP non sia attivata l'opzione –enable-trans-sid che lascia a PHP il compito di inserire nei nostri link relativi l'id di sessione. L'assenza dell'id di sessione fa credere a session_start() che si stia iniziando una sessione nuova, e quindi non ci permetterebbe di recuperare i valori salvati e continuare la nostra sessione.

Io userò due variabili di sessione:

- **$_SESSION['user']:** memorizza l'username dell'utente che ha effettuato il login. Verrà registrata quindi nel momento in cui un utente vienne autenticato mentre verrà cancellata nel caso in cui si effettua il logout

- **$_SESSION['info']:** questa variabile invece tiene in memoria i dati relativi alla lezione (tutti racchiusi in una singola stringa) generati ed inviati dalla pressione del tasto in Flash presente nel sistema Lode in un istante qualsiasi della lezione che si sta visualizzando. Solo in questo modo un utente può inserire un commento ed è grazie a questa variabile se si riesce a capire in che modo l'utente è arrivato al sistema. La variabile verrà registrata nel momento in cui arrivano i dati da Lode mentre verrà distrutta quando l'utente ha inserito il proprio commento oppure ha effettuato il logout.

4.3.6 Usiamo PHP per accedere al database

E' arrivato il momento di capire come si possa accedere al database attraverso uno script php. Questo ovviamente ci permetterà di eseguire qualsiasi operazione sul database. In particolare dovremmo poter eseguire query di inserimento e di ricerca.

Prima di poter fare ricerche però, dobbiamo collegarci al database e per fare questo abbiamo bisogno di alcune informazioni e parametri del database che solamente chi gestisce il sitema potrà sapere:

```
$db_host = "localhost";
$db_user = "gabriele";
$db_password = "prova";
$db_name = "dbprova";
```

A questo punto abbiamo tutti i dati necessari per la connessione al database. Questa la possiamo realizzare attraverso la funzione mysql_connect:

```
$con = mysql_connect($db_host, $db_user, $db_password);
```

Se la connessione ha buon fine ci restituisce un identificatore alla connessione che noi memorizziamo in $con. Questa variabile la utilizzeremo ogni volta che vorremo fare un'operazione sul database. Se la connessione non dovesse andare a buon fine (per esempio se uno dei parametri fosse sbagliato) verrebbe restituito FALSE.

Fatto tutto questo dobbiamo specificare su quale database vogliamo lavorare:

```
mysql_select_db($db_name, $con);
```

In questo caso non abbiamo bisogno di memorizzare alcun valore, visto che la funzione restituisce solo TRUE o FALSE.

Ora è veramente possibile eseguire una qualsiasi query. Pagina per pagina esamineremo in seguito le query utilizzate nel sistema. Vediamo ora come possiamo comunicarle al database. Ebbene ciò è possibile con il seguente comando:

mysql_query($query, $con)

dove $con è l'identificatore della connessione al database e $query è una stringa contenente la query da porre al database.

Se la query è di ricerca allora questa ci restituisce un identificatore che si può usare in seguito per andare ad osservare nel dettaglio i risultati.:

```
$result=mysql_query($querySearch, $con)
```

Quando il database non è più utile allo script è consigliabile (ed è quello che ho fatto ogni volta) chiudere la connessione con il seguente comando:

```
mysql_close($con);
```

4.3.7 Usiamo PHP per stampare le pagine

Come illustrato in precedenza tutte le pagine presenti hanno la stessa struttura. Questo ci permette di definire due funzioni:

print_inizioPag($titolo,$sottotitolo)

Questa è la funzione che stampa la pagina fino a:

```
<div id="BoxPrincipale"><div class='TitoloBoxPrincipale'></div>
```

E' infatti da questo punto che parte la sezione che contiene tutto ciò che interessa all'utente (form per ricercare, inserire commenti, registrarsi, loggarsi, risultati di ricerche o visualizzazione di messaggi d'errore) e che viene modificata a seconda della pagina in cui ci si trova.

Come avete potuto vedere in precedenza il Box Principale è provvisto di un titolo e di un sottotitolo. Questi vengono passati alla funzione che li stampa nella sezione apposita.

Inoltre, a seconda del titolo, la funzione riconosce che tipo di contenuti ci saranno nel box principale e si adeguerà di conseguenza assegnando stili diversi a seconda del titolo (es. se ci troviamo davanti ad un errore verrà visualizzata la scritta "ERRORE" su sfondo rosso)

La funzione ha il dovere di stampare anche il menu della pagina. Dato che però questo, come già esposto, visualizza solo i link disponibili a seconda dello stato di ogni utente collegato, è necessario scrivere le seguenti linee di codice che utilizzano le variabili di sessione:

```
<div id="Menu">
<ul>
<?php
$username=$_SESSION['user'];
if($username=="")
    print "<li class='prima'>MENU</li>";
else
    print "<li class='prima'>CIAO $username</li>";
?>
<li class="page_item"><a href="home.php" class="MenuLode">
   <span class="style1">HOME</a></li>
<?php
if(!IsSet($_SESSION['user'])) {
```

```
    print "<li class='page_item'><a href='login.php'
           class='MenuLode'>LOGIN</a></li>" ;
    print "<li class='page_item'><a href='register.php'
           class='MenuLode'>REGISTRAZIONE</a></li>" ;
}
else
{
if ((IsSet($_SESSION['info'])) && ($_SESSION['info']!=""))
    print "<li class='page_item'><a href='login.php'
           class='MenuLode'>SCRIVI IL COMMENTO</a></li>";
    print "<li class='page_item'><a href='search.php'
           class='MenuLode'>RICERCA COMMENTI</a></li>" ;
    print "<li class='page_item'><a href='logout.php?logout=1'
           class='MenuLode'>LOGOUT</a></li>" ;
}
?>
<li class="page_item"><a href="#" class="MenuLode">LINK</a></li>
</ul>
</div>
```

Come si può vedere a seconda delle variabili di sessione si scrivono o meno vari link.

Partiamo all'inizio del codice nel quale, se è settata la variabile di sessione 'user' (cioè un utente ha effettuato il login al sistema) viene stampata la stringa "CIAO nomeutente" altrimenti viene stampato "MENU". Nello stesso modo vengono stampati nel menu i link alle pagine di ricerca, inserimento di un commento, logout se l'utente è loggato altrimenti verranno stampati i link alle pagine di login e registrazione.

Nel codice c'è un'altra variabile di sessione definita sull'indice 'info'. Questa è settata solamente se si arriva al sistema attraverso il pulsante in flash presente nella lezione che si stava visualizzando grazie a Lode. Un commento può essere scritto solamente se questo accade e quindi il link SCRIVI IL COMENTO sarà visibile solamente in quel caso.

print_finePag()

Qquesta funzione, come dice il nome stesso, stampa la parte finale della pagina: dal che chiude la parte del BoxPrincipale aperta nella funzione precedente fino ad </html> che chiude la pagina. Si trova anche una exit(); in fondo che fa in modo di terminare lo script in quell'istante.

Queste funzioni verranno chiamate in tutti gli script php relativi a tutte le pagine del sistema. Il grosso vantaggio è sicuramente un risparmio di memoria ma soprattutto la possibilità di modificare la struttura di tutto il sistema cambiando solo la funzione interessata senza agire su ogni pagina.

Ora andrò ad esaminare nel dettaglio le pagine... per motivi di spazio non riporto tutto il codice presente nei vari script... in particolare mancheranno i dettagli HTML, le chiamate alle funzioni appena descritte, le operazioni di apertura e chiusura della connessione al database e tutti i controlli riguardanti l'esecuzione corretta di una query. Tutte queste operazioni sono state descritte in precedenza. L'importante è capire la logica di funzionamento.

4.3.8 login

Per poter effettuare qualsiasi operazione (inserimento di commenti, ricerche, modifiche di commenti) è necessario effettuare il login al sistema con username e password specificati in fase di registrazione.

La pagina quindi:

- permette invio di username e password;
- controlla presenza di username nella tabella utenti e l'uguaglianza della password
- reindirizza l'utente alla pagina dedicata all'inserimento del commento nel caso in cui siano arrivati dati dal sistema Lode (pressione del pulsante apposito in Flash)

Tutto ciò è gestito dallo script login.php.

Il form per l'immissione dei dati da parte dell'utente è il seguente:

```
<form method="post" action="login.php">
USERNAME:<INPUT TYPE=TEXT SIZE=20 NAME=posted_username>
PASSWORD:<INPUT TYPE=PASSWORD SIZE=20 NAME=posted_password>
<input name="SUBMIT" value="Loggami" type="submit">
</form>
```

Descrivo in seguito le principali righe di codice.

Inizialmente lo script si occupa della gestione dei dati inviati dall'utente attraverso il form:

```
/*registriamo la variabile di sessione $_SESSION['info'] solo
  se sono arrivati dati da Lode*/
if (IsSet($_GET['info']))
    $_SESSION['info']=$_GET['info'];
//se sono stati inviati i dati dal form di login
if (IsSet($_POST['posted_username']) &&
    IsSet($_POST['posted_password'])) {
    //salviamo username inviato dal form
    $us=$_POST['posted_username'];
    //ci connettiamo al database interessato
```

```
    ...
    /*estrapoliamo dalla tabella degli utenti registrati
      l'elemento corrispondente all'username inviato */
    $sql="SELECT * FROM utenti WHERE username='$us'";
    $rigo = mysql_query($sql,$con);
    //se ha trovato un elemento
    if (mysql_num_rows($rigo) == 1) {
        /*salvo login e password dell'utente registrato
          nel database...*/
        $login_user = mysql_result($rigo,0,"username");
        $pass_user = mysql_result($rigo,0,"password");
        /*... e li controllo con quelli inviati attraverso il
          form dall'utente*/
        if($login_user==($_POST['posted_username']) &&
           $pass_user==md5($_POST['posted_password'])) {


           /*se sono uguali creo la variabile di sessione...
             l'utente è loggato*/
           $_SESSION['user']=$_POST['posted_username'];
        }
        else
           $errore=1;
           //poi stamperemo l'errore "login o password errati"
        }
    }
    else
        $errore=1;
        //poi stamperemo l'errore "login o password errati"
}
```

La sezione successiva controlla se siamo riusciti ad accedere al sistema effettuando correttamente il login:

```
//se l'utente non è loggato
if(!IsSet($_SESSION['user'])) {
    //scriviamo il form per l'immissione dei dati
    //ecco qui la stampa dell'errore
    if ($errore==1)
        print "ERRORE: Login o password errati!!!";
}
else
```

```
    //siamo loggati pagina riservata
    //se siamo atrrivati qui dal pulsante Flash della lezione
    if (IsSet($_SESSION['info'])&& $_SESSION['info']!="")

        // indirizzati alla pagina di inserimento del commento
        header("location:login_success.php");
    else {
        //altrimenti viene stampato un messaggio
        $us=$_SESSION['user'];
        print "Ciao $us Login effettuato correttamente!!";
    }
?>
```

4.3.9 logout

Un utente loggato correttamente deve avere anche la possibilità di effettuare il logout ad esempio per poter accedere con un altro username.

Il modo corretto per effettuare il logout è quello di premere il link presente nel menu:

```
<li class='page_item'><a href='logout.php?logout=1'
class='MenuLode'>LOGOUT</a></li>" ;
```

come potete vedere il dato logout=1 viene accodato all'Url. In questo modo questo viene inviato con il metodo GET senza utilizzo di form.

E' lo scipt più breve e semplice:

```
//se l'utente non è loggato
if(!IsSet($_SESSION['user']))
    print "ERRORE Devi prima effettuare il login!";
else
    //se siamo arrivati a questa pagina in modo corretto
    if($_GET['logout']==1) {
        // distrugge la sessione
        $_SESSION=array();
        session_destroy();
        print "Logout avvenuto con successo!!";
    }
    else

        print "Sei arrivato in questa pagina in modo errato!";
```

4.3.10 registrazione

Per accedere ai servizi offerti dal sistema è necessario registrarsi. La pagina che permette di fare ciò è generata dallo script register.php che vediamo nel dettaglio fra un attimo.

I dati vengono immessi nel seguente form:

```
<form method="post" action="register.php">
<input name="attivazione" value="2" type="hidden">
USERNAME:<input name="posted_username" type="text">
PASSWORD:<input name="posted_password" type="password">
RIPETI PASSWORD:<input name="posted_rep_pass" type="password">
NOME:<input name="posted_name" type="text">
COGNOME:<input name="posted_surname" type="text">
<input name="SUBMIT" value="Registrami" type="submit">
</form>
```

Come si può osservare, i dati vengono inviati allo script (register.php) stesso che quindi può essre richiamato dall'apposito link presente nel menu oppure cliccando sul tasto Registrami presente nel form. Il tutto deve essere ovviamente gestito, vedremo anche questo nella seguente descrizione dettagliata del file register.php.

Inizialmente si salvano i dati inviati attraverso il form visto in precdenza. In realtà i dati si memorizzano solo se attivazione=2, cioè se siamo arrivati alla pagina attraverso la pressione del pulsante presente nel form e quindi inviando i dati. Infatti se la pagina è stata richiamata attraverso il menu attivarzione non sarà 2 e non sarà ovviamente necessario salvare tutti gli altri dati in quanto non presenti:

```
$attivazione=$_POST['attivazione'];
if ($attivazione=='2') {
    $username=$_POST['posted_username'];
    $pass=$_POST['posted_password'];
    $name=$_POST['posted_name'];
    $surname=$_POST['posted_surname'];
    $rep_pass=$_POST['posted_rep_pass'];
}
```

Poi si controlla se siamo già loggati. Ho previsto che non si debba esserlo per poter effettuare una registrazione.

```
if (IsSet($_SESSION['user']))

    print "Prima di fare una registrazione devi sloggarti!!";
```

Poi c'è la parte che verifica i dati inviati:

```
//se sono arrivato alla pagina tramite invio dei dati
if ($attivazione=='2'){
```

```
//mi connetto al database
...
/*lo interrogo per vedere se ci sono altri utenti con
  l'username che desidero utilizzare*/
$sql="SELECT * FROM utenti WHERE username='$username'";
$rigo = mysql_query($sql,$con);
//se ci sono già username
if (mysql_num_rows($rigo) != 0) {
    //allora stampo un messaggio e riscrivo il form
    echo "ERRORE: esiste già' $username come username!!";
    /*qui non faccio altro che stampare il form descritto
      in precedenza magari tenendo i valori di nome e
      cognome inviati in precedenza:
      es. <input name="posted_name" type="text"
           value="<?php print $name;?>">*/
}
else
    //username può andare
    //qui faccio la validazione del form
    if (($username!="") && ($pass!="") && ($name!="") &&
       ($surname!="") && ($rep_pass!="")){
       //se le password inserite sono diverse
       if ($pass!=$rep_pass) {
         echo "ERRORE: PASSWORD DIVERSE";
         /*qui non faccio altro che stampare il form
           descritto in precedenza magari tenendo i valori
           di username, nome e cognome inviati prima*/
       }
       else{
         $sql="INSERT INTO utenti
                (username ,name ,surname ,password)
              VALUES
              ('$username','$name','$surname',md5('$pass'))";
         mysql_query($sql,$con);
         echo "Registrazione effettuata correttamente!!!";
         mysql_close($con);
       }
    }
    //qui arrivo se ho lasciato qualche campo vuoto
    else {

       echo "Form non completato correttamente!!!";
       /*qui non faccio altro che stampare il form descritto
```

```
            in precedenza tenendo i valori inseriti ed evidenziando
            i campi lasciati vuoti:
            es. <?php
              if ($username==""){echo"<font color=\"red\">";}?>
              USERNAME:<?php if($username==""){echo"</font>";}?>
              <input name="posted_username" type="text"
              value="<?php print $username;?>" size="20">*/
        }
}
/*in questo caso siamo arrivati alla pagina senza aver inviato
i dati quindi scriviamo semplicemente il form */
else {
    //qui stampiamo il form
}
```

4.3.11 inseriamo un commento

E' qui il fulcro del sistema. Lo script login_success.php al quale si viene reindirizzati nello script login.php gestisce l'inserimento di un commento.

Il form nel quale sarà possibile scrivere ciò che si desidera è il seguente:

```
<form name="form1" method="post" action="login_success.php">
<div align="center">
<?php
$oFCKeditor->BasePath = '../fckeditor/' ;
$oFCKeditor = new FCKeditor('FCKeditor1') ;
$oFCKeditor->BasePath = '../fckeditor/' ;
$oFCKeditor->name = 'fckeditor' ;
$oFCKeditor->Value = $comm ;
$oFCKeditor->Height = '400' ;
$oFCKeditor->Create() ;
?>
</form>
```

E' il form più semplice di tutte le pagine. Invia solo un dato: il commento (Per dubbi riguardanti FCKEditor riguardare il paragrafo 4.3.4). L'attributo ACTION indica che la pagina che processerà questa informazione è proprio login_success.php alla quale si può quindi accedere correttamente in tre modi: attraverso il menu, attraverso il reindirizzamento presente in login.php visto prima oppure attraverso l'invio dei dati da parte del form generato da login_success.php. Questo aspetto è da gestire come anche il fatto di permettere di inserire un commento solo a chi è arrivato al sito internet attraverso il pulsante flash del sistema Lode con conseguente invio di varie informazioni.

Inoltre il database è fatto in modo da popolarsi automaticamente. Se vengono infatti mandati dati riguardanti una lezione non presente nel database, allora lo script dovrà

aggiungere, oltre all'elemento nella tabella commenti, anche una lezione con tutti i suoi attributi nella tabella Lezione appunto.

Prima di vedere lo script esaminiamo la funzione smembraInfo($info). Tale funzione serve per dividere i dati che arrivano da Lode tramite il pulsante in Flash su una singola stringa in un array. A tale funzione viene sempre passato il valore della variabile di sessione $_SESSION['info'], una stringa che ha sempre la seguente struttura:

```
id=test;corso=test;numLezione=intero;slide=test;
  secondi=intero;speaker=test
```

Ecco la funzione:

```
function smembraInfo ($info)
    {
    $a= strpos($info,"id=");
    $b= strpos($info,";corso=");
    $c= strpos($info,";numLezione=");
    $d= strpos($info,";slide=");
    $e= strpos($info,";secondi=");
    $f= strpos($info,";speaker=");
    $infor[0] = substr($info,$a+3,$b-$a-3);
    $infor[1] = substr($info,$b+7,$c-$b-7);
    $infor[2] = substr($info,$c+12,$d-$c-12);
    $infor[3] = substr($info,$d+7,$e-$d-7);
    $infor[4] = substr($info,$e+9,$f-$e-9);
    $infor[5] = substr($info,$f+9);
    $infor[6]=strpos($stringa,";");
    return $infor;
}
```

Ecco lo script nelle sue righe più significative:

Inizialmente controlliamo se l'utente è loggato. In caso contrario lo reindirizziamo alla pagina di login

```
if(!IsSet($_SESSION['user']))
    header("location:login.php");
```

Successivamente, se loggati, si controlla che l'utente sia arrivato qui attraverso il sistema Lode con conseguente invio dell'informazione. In questo caso la pagina viene chiamata da login.php che prima però salva la sessione con indice 'info'.

```
if ((!IsSet($_SESSION['info'])) || ($_SESSION['info']==""))

    print("Sei arrivato a questa pagina in modo errato...");
```

A questo punto si verifica se è già stato inviato il commento dal form (ricordo che il form invia i dati alla pagina stessa). In questo caso si gestisce l'inserimento nel database altrimenti si stampa la pagina che servirà all'inserimento

```
if (!IsSet($_POST[FCKeditor1])) {
    //nessun commento è stato ancora inviato
    //creo la pagina per scrivere il commento
    $inf=smembraInfo($_SESSION['info']);
    $username=$_SESSION['user'];
    //stampiamo a video le informazioni riguardanti la lezione
    print "
    Corso $inf[1]
    Lezione $inf[2]
    Argomenti $inf[3]
    Professore $inf[5]
    Secondi $inf[4]
    Commento
    //stampo il form per l'inserimento del commento
}
else { //abbiamo già scritto e inviato il commento
    $inf=smembraInfo($_SESSION['info']);
    $username=$_SESSION['user'];
    //guardiamo se la lezione è già presente nel database
    $sql="SELECT * FROM `lezione` WHERE `id_lezione` = '$inf[0]'";
    $rigo = mysql_query($sql,$con);
    //se non è già memorizzata
    if (mysql_num_rows($rigo)==0){
        //la inseriamo
        $sql="INSERT INTO lezione
          (id_lezione ,lezione ,corso ,slide,speaker)
          VALUES
          ('$inf[0]','$inf[2]','$inf[1]' ,'$inf[3]','$inf[5]')";
        mysql_query($sql,$con);
        print "Lezione non presente ma inserita correttamente!";
    }
    //inseriamo il comemnto nel database
    $sql="INSERT INTO commento
      (username ,id_lezione ,time_secondi ,commento)
      VALUES
      ('$username','$inf[0]','$inf[4]','$_POST[FCKeditor1]')";
    mysql_query($sql,$con);
    print "Commento inserito correttamente!!!";
    /*desettiamo la sessione 'info' in quanto il commento è
      stato inserito e non deve esserci la possibilità per
```

```
        l'utente di inserire un nuovo commento senza cliccare il
        pulsante in Flash di Lode*/
    unset ($_SESSION['info']);
}
```

4.3.12 ricerchiamo un commento

Il sistema offre la possibilità di ricercare alcuni commenti attraverso diversi campi quali: nome del corso, numero della lezione, professore che la tiene, titolo della slide, secondi del video della lezione fra i quali è stato scritto il commento, autore e parola chiave da ricercare all'interno del testo dello stesso. Se i commenti trovati appartengono alla persona che in quel momento è loggata al sistema allora deve esserci la possibilità di poter modificare la propria opinione aggiungendo un link che porterà alla pagina di modifica del documento.

Lo script che gestisce il tutto è racchiuso nel file search.php che per molti aspetti è molto simile al register.php. Iniziamo col form principale nel quale inserire i vari campi di ricerca:

```
<FORM METHOD=GET ACTION=search.php>
<input type=hidden name=attivazione value=2>
CORSO:<INPUT TYPE=TEXT SIZE=20 NAME=corso>
NUMERO LEZIONE:<INPUT TYPE=TEXT SIZE=5 NAME=nLez>
TITOLO SLIDE:<INPUT TYPE=TEXT SIZE=20 NAME=slide>
PROFESSORE:<INPUT TYPE=TEXT SIZE=20 NAME=prof>
SECONDI DA: <INPUT TYPE=TEXT SIZE=5 NAME=secDa VALUE='0'>
A:<INPUT TYPE=TEXT SIZE=5 NAME=secA VALUE='100000'>
AUTORE COMMENTO:<INPUT TYPE=TEXT SIZE=20 NAME=autore>
STRINGA DA RICERCARE:<INPUT TYPE=TEXT SIZE=20 NAME=stringa>
<INPUT TYPE=SUBMIT NAME=SUBMIT VALUE=Ricerca>
</FORM>
```

Anche in questo caso il form ha come valore di ACTION lo script stesso che quindi può essere eseguito in caso di click sull'apposito link del menu oppure sul pulsante presente nel form.

Ecco lo script nelle parti più significative:

Inizialmente viene salvato il dato $attivazione che funziona come in Register.php. Se necessario quindi vengono salvati tutti gli altri dati (solo se $attivazione=2)

```
$attivazione=$_GET['attivazione'];
    if ($attivazione=='2') {
    $stringa=$_GET['stringa'];
    $corso=$_GET['corso'];
    $nLez=$_GET['nLez'];
    $slide=$_GET['slide'];
    $prof=$_GET['prof'];
    $secDa=$_GET['secDa'];
```

```
    $secA=$_GET['secA'];
    $autore=$_GET['autore'];
}
```

Poi c'è il solito controllo sulla variabile di sessione per vedere se si è loggati ed in caso contrario veniamo reindirizzati alla pagina di login:

```
if(!IsSet($_SESSION['user']))
    header("location:login.php");
```

Syccessivamente si iniziano a controllare i dati ed in caso di errore si imposta la variabile problema a 1... se non ci sono problemi di validazione allora si può effettuare la ricerca.

```
//se sono stati inviati i dati
if ($attivazione=='2'){
    // azzero la variabile problema
    $problema=0;
    //se i secondi inviati non sono interi oppure sono null
    if ((!is_num_or_null($secDa))||(!is_num_or_null($secA)))
        $problema=1;
    else

        //se secondi di partenza maggiori dei secondi di fine
        if ($secDa>$secA)
            $problema=1;
    //se non inviato o non intero il numero della lezione
    if (!is_num_or_null($nLez))
        $problema=1;
    //se i dati sono validi
    if ($problema==0) {
        //mi connetto al database
        ...
        //ecco la query si ricerca
        $sql="SELECT * FROM commento, lezione
          WHERE ((commento.id_lezione=lezione.id_lezione) &&
          (commento LIKE '%$stringa%') &&
          (speaker LIKE '%$prof%') &&";
        if ($nLez!="")
            $sql="$sql( lezione = $nLez ) &&";
        $sql="$sql( slide LIKE '%$slide%' ) &&
          (username LIKE '%$autore%') &&
          (corso LIKE '%$corso%') &&
          (( time_secondi >= $secDa ) &&
```

```
        (time_secondi <= $secA)))";
      //eseguo la query
      $rigo = mysql_query($sql,$con);
      //memorizzo il numero di elementi trovati
      $ristot=mysql_num_rows($rigo);
      //se non ci sono elementi trovati esco dallo script
      IF ($ristot==0){

         print "Non è stato trovato alcun commento!!!";
         exit();
      }
      /*ci sono elementi trovati quindi eseguo un ciclo che
        me li visualizza tutti*/
      while ($rigo2=mysql_fetch_array($rigo)){
         $comm = $rigo2 ["commento"];
         $scritto = $rigo2 ["username"];
         $nlex = $rigo2 ["lezione"];
         $corso = $rigo2 ["corso"];
         $slide = $rigo2 ["slide"];
         $speaker = $rigo2 ["speaker"];
         $sec = $rigo2 ["time_secondi"];
         $idComm=$rigo2 ["id_commento"];
         echo "SCRITTO DA: $scritto";
         echo "CORSO: $corso";
         echo "LEZIONE: $nlex";
         echo "ARGOMENTO: $slide";
         echo "PROFESSORE: $speaker";
         echo "SECONDI: $sec";
         echo "COMMEMTO: $comm";
         /*qui si scrive il link che consente la modifica del
           commento solo se autore del commento trovato e
           utente loggato sono uguali*/
         if ($scritto==$_SESSION['user'])
           print("
             <A HREF=\"modCommento.php?idCommento=$idComm\">
             Modifica il commento</A>");
      }
   }
   else { //qui andiamo se problema=1
   /*qui non faccio altro che stampare il form descritto in
   precedenza tenendo i valori inseriti ed evidenziando i
   campi lasciati vuoti o errati*/
   }
}
```

```
else
{
//qui arrivo se reindirizzato a questa pagina con il menu
//stampo il form precedente
}
```

4.3.13 modifichiamo un commento

E' lo script modCommento.php che si occupa di questa funzionalità.

Anche in questo caso ci troviamo di fronte ad un form che è molto simile a quello presente nella pagina di inserimento di un commento a parte l'intestazione, il nome del pulsante e il dato idCommento che nell'altro non è presente:

```
<form name="form1" method="get" action="modCommento.php">
<div align="center">
<?php
$oFCKeditor->BasePath = '../fckeditor/' ;
$oFCKeditor = new FCKeditor('FCKeditor1') ;
$oFCKeditor->BasePath = '../fckeditor/' ;
$oFCKeditor->name = 'fckeditor' ;
$oFCKeditor->Value = $comm ;
$oFCKeditor->Height = '400' ;
$oFCKeditor->Create() ;
print "<input type=hidden name=idCommento value=$idCommento>";
?>
<input type="submit" value="MODIFICA" name="B1">
</form>
```

Il form invia due dati: idCommento e il commento (Per dubbi riguardanti FCKeditor riguardare il patagrafo 4.3.4). Anche in questo caso i dati vengono mandati allo stesso script.

Come spiegato prima solo l'autore può modificare un dato commento e quindi lo script dovrà controllare che questo accada. modCommento.php può essere richiamata dal form di invio dati oppure dal link presente vicino ad un commento trovato nella pagina di ricerca. Andiamo ad esaminare questo link:

<A HREF=\"modCommento.php?idCommento=$idComm\">Modifica il commento</A>

Come si può osservare viene richiamata la pagina con concatenato l'id del commento che verrà modificato, id che risulta necessario allo script per identificare il commento in questione. Un malintenzionato potrebbe però richiamare direttamente la pagina con concatentato l'id che identifica un commento di cui non è l'autore. E' ovvio che lo script dovrà prevedere questo e non consentire che ciò accada.

Andiamo come sempre nel dettaglio:

```
/*Se non siamo loggati o se non è stato inviato l'id commento
  (es. se andiamo nel browser ed inseriamo nella barra degli
  indirizzi modCommento.php direttamente)*/
if(!IsSet($_SESSION['user'])||!IsSet($_GET['idCommento']))

    //veniamo reindirizzati alla pagina di login.
    header("location:login.php");

//parte privata
//effettuiamo la connessione al database
...
//memorizziamo l'identificatore del commento in una variabile
$idCommento=$_GET['idCommento'];
/*ecco la query sql che estrai i dati relativi al commento che
  desideriamo modificare*/
$sql="SELECT * FROM commento, lezione
      WHERE ((commento.id_lezione = lezione.id_lezione)
            && (`id_commento` = '$idCommento'))";
//eseguiamo la query
$rigo = mysql_query($sql,$con);
/*salviamo il primo e unico (se c'è, dato che l'id è la chiave
  primaria) risultato della query*/
$rigo2=mysql_fetch_array($rigo);
//salviamo i campi dell'elemento trovato che ci interessano
$comm = $rigo2["commento"];
$scritto = $rigo2["username"];
$nlex = $rigo2 ["lezione"];
$corso = $rigo2 ["corso"];
$slide = $rigo2 ["slide"];
$speaker = $rigo2 ["speaker"];
$sec = $rigo2 ["time_secondi"];
/*ecco il controllo sull'uguaglianza tra l'autore del commento
  estrapolato e l'utente loggato*/
if ($scritto!=$_SESSION['user']){

    print "ERRORE: non sei l'autore del commento...
           non puoi modificarlo!!!";
    exit();

}
//... ok! L'autore corrisponde
/*controlliamo se siamo arrivati a questa pagina tramite il
  link nella pagina di rERRORE: icerca*/
if (!IsSet($_GET[FCKeditor1])) {


    //scriviamo i dati non editabili riguardanti il commento
    echo "SCRITTO DA: $scritto";
```

```
    echo "CORSO: $corso";
    echo "LEZIONE: $nlex";
    echo "ARGOMENTO: $slide";
    echo "PROFESSORE: $speaker";
    echo "SECONDI: $sec";
    print "COMMENTO";
    /*qui si inserisce il form visto prima che consente di
    modificare il commento*/
}
else {

    /*siamo arrivati qui attraverso il form con relativo invio
      del commento modificato*/
    //aggiorniamo il commento nel database
    $username=$_SESSION['user'];
    $sql="UPDATE commento SET commento = '$_GET[FCKeditor1]'
          WHERE `id_commento` = $idCommento";
    mysql_query($sql,$con);
    print "Commento modificato correttamente!!";
}
```

4.4 Altri metodi per fare annotazione

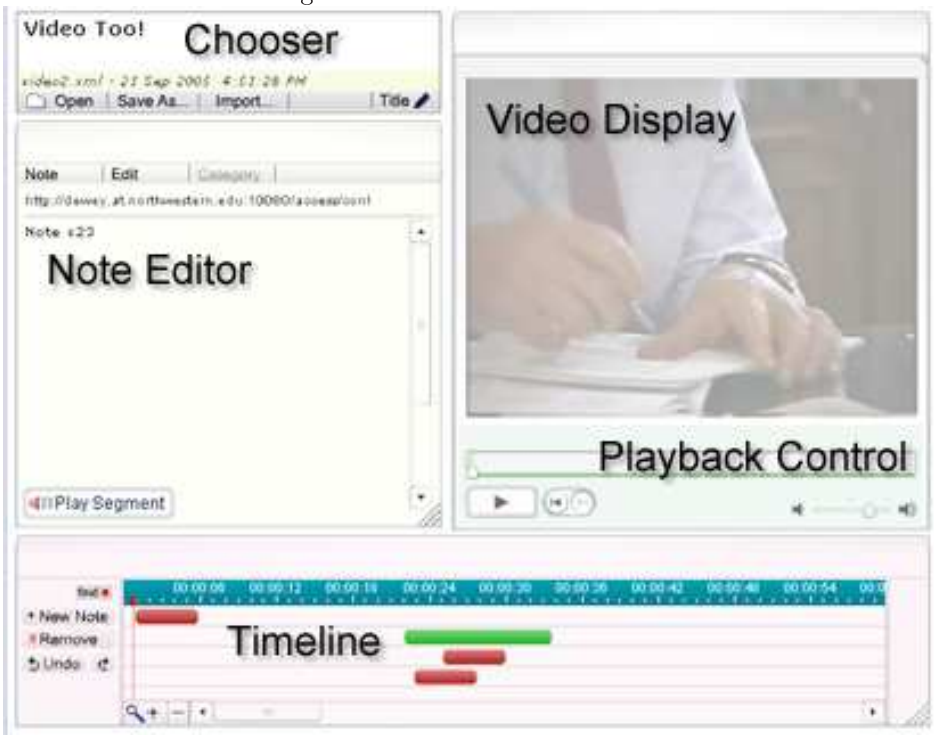
4.4.1 Introduzione

Navigando in rete si possoo trovare vari sistemi di annotazione che hanno lo stesso obiettivo del sistema da me progettato: dare la possibilità di scrivere note e commenti ad istanti video ben definiti. Di seguito riporto tre esempi si software che offrono queste funzionalità. Infine ritengo giusto dover fare una breve descrizione dell'MPEG7: standard nato per definire come sono organizzati i dati multimediali. Scopriremo perchè non sia possibile utilizzare questo per il nostro scopo.

4.4.2 Project Pad

Project Pad è uno strumento molto sofisticato che include strumenti per annotare video, audio e persino immagini.

Figura 9: Video Annotation Tool



L'annotazione video è possibile grazie al Video Annotation Tool che permette di aggungere note a precise parti di video. Il tutto è possibile attraverso una gradevole interfaccia che vedete in figura 9 molto simile a quella di Lode Flash divisa in cinque parti principali:

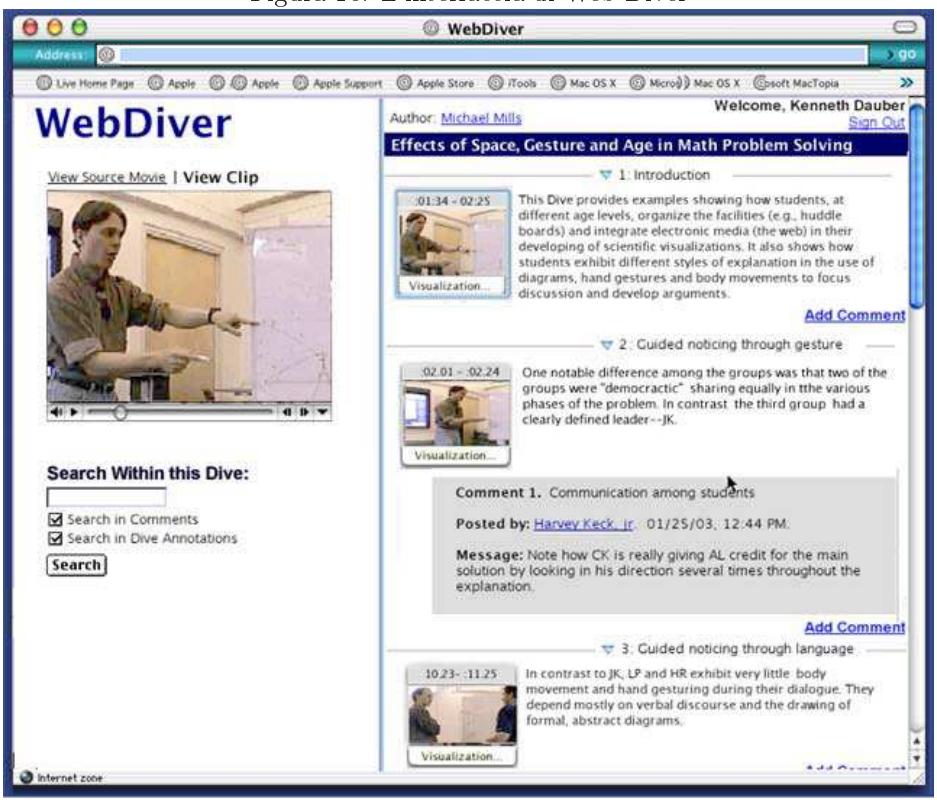
1. **Chooser**: si può aprire un'annotazione, cambiare il titolo o salvare una copia di essa;
2. **Playback Control**: qui c'è la possibilità di mandare in play/pause il video e controllare il livello audio. C'è anche una barra per vedere a che punto siamo del video.
3. **Note editor**: in questa parte si modifica il testo dell'annotazione.
4. **Timeline**: visualizza il tempo corrente del video e i segmenti in cui sono presenti annotazioni. Ci sono anche i controlli per aggiungere o rimuovere le note.
5. **Video Display**: si occupa della visualizzazione del video.

In questo sistema annotare è molto semplice: basterà portarsi nel punto in cui si vuole scrivere una nota e cliccare su un apposito pulsante. La nota può essere associata ad

un segmento di timeline ed il sistema gestisce questo in modo molto efficace con appositi strumenti: aggiungere, visualizzare, rimuovere una nota oppure modificare il punto iniziale e finale di un segmento è veramente molto semplice ed intuitivo.

4.4.3 Web Diver

Figura 10: L'interfaccia di Web Diver



Un altro esempio è Web Diver (figura 10), sistema progettato e realizzato dai ricercatori della Stanford University (CA – USA) presso lo SCIL (Stanford Centre for Innovation in Learning), con lo scopo di mettere in luce i vantaggi e le potenzialità didattiche dell'uso della segmentazione ed annotazione di filmati digitali in rete. Questo sistema infatti è facilmente usabile anche da chi ha competenze e conoscenze limitate nel campo dell'informatica in genere e negli scambi comunicativi in rete.

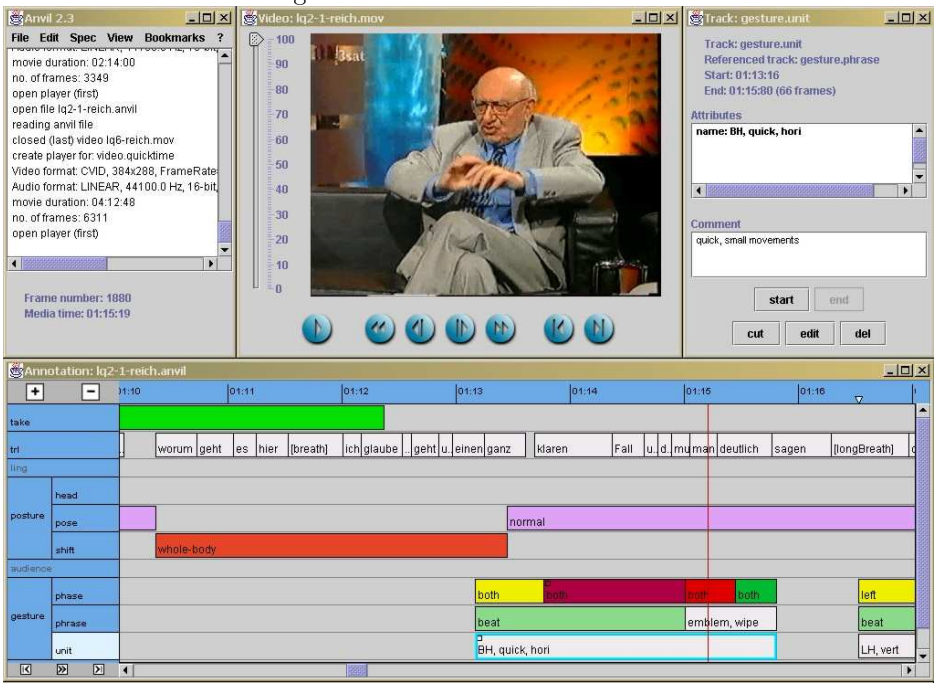
Con poche semplici funzionalità permette agli utenti di realizzare e condividere percorsi di approfondimento e riflessione, utili a generare la consapevolezza delle tappe seguite e dei progressi ottenuti, migliorando l'apprendimento e favorendo l'interiorizzazione di processi e contenuti. Il sistema Web Diver è una tecnologia web video digitale avanzata che include strumenti per la selezione di singole scene video tratte da un filmato esistente

al fine di effettuare una analisi collaborativa delle scene stesse. Esso si basa sull'idea che l'utente, immergendosi nei filmati, possa creare nuovi punti di vista e commentarli scrivendo brevi paragrafi di testo. Il sistema rende possibile creare rapidamente una infinita varietà di nuovi video clips digitali (dives) da qualunque registrazione video. Il prodotto finale è quindi una collezione di brevi segmenti video, separati tra loro, che rappresentano il punto di vista dell'utente su aspetti particolari del filmato stesso. Questi segmenti possono essere condivisi su internet tra insegnanti, tra studenti e studenti, tra insegnanti e studenti o tra colleghi, in scenari diversi dagli ambienti di apprendimento.

4.4.4 Anvil

In figura 11 potete osservare come si presenta questo sistema a qualsiasi utente.

Figura 11: L'interfaccia di ANVIL



Anche in questo caso la finestra è divisa in parti ben distinte:

1. **Main Window**: è la prima finestra che si vede all'apertura di Anvil. Contiene menu e pulsanti che permettono di accedere alle funzioni vitali del sistema e visualizza varie informazioni sul video e sulle annotazioni.

2. **Annotation Board**: è il cuore del sistema, disposta in basso, è una timeline. La linea rossa verticale identifica la posizione corrente del video, l'utente può disporsi al

punto desiderato agendo su questa. Qui vengono creati gli oggetti per l'annotazione, questi sono ben visibili e facili da trovare.

3. **Element Window**: qui vengono visualizzate le informazioni riguardanti l'oggetto: attributi, valori e commento.

4. **Element Edit Window**: è usata quando un utente vuole creare un nuovo o modificare un oggetto già esistente. La sezione offre strumenti molto interessanti per agevolare chiunque fosse interessato.

5. **Video Window**: posto nella parte superiore, in mezzo alla finestra, visualizza il video e alcuni pulsanti di gestione: play/pause, forward/backward.

Anvil consente un'annotazione gerarchica multi-livello con "oggetti" definibili dall'utente e, di conseguenza, di natura arbitraria. La visualizzazione sincronizzata delle immagini video con le annotazioni rendono la codifica assai intuitiva. Sono consentiti legami fra livelli ed è stato realizzato un "project manager" di ausilio alle procedure di ricerca. Anvil è scritto in Java e utilizza il linguaggio XML per la memorizzazione dei dati.

4.4.5 MPEG-7

MPEG-7 è uno standard nato per la descrizione dei contenuti multimediali oggi presenti in ogni ambito della comunicazione.

Formalmente chiamato "Multimedia Content Description Interface", si pone nell'ottica di fornire strumenti e standard per la descrizione di dati di tipo multimediale (o AV: audio-video), non considerando la loro particolare tecnologia di memorizzazione e trasmissione, ma ponendosi come valido strumento per la gestione e interpretazione del contenuto informativo presente in essi. Con una espressione inglese si potrebbe dire che riguarda "bits about the bits", ossia dati riguardanti dati, che, con termine tecnico, si potrebbero descrivere come meta-dati.

Fine ultimo dell' MPEG-7 è quindi, in uno scenario futuro che vedrà sempre più presente internet e il web nella vita di tutti i giorni, far si che il web sia "esplorabile" per quanto riguarda i contenuti multimediali, così come oggi lo è per documenti di tipo testuale.

Per un media di tipo visivo come un filmato si può quindi spaziare da astrazioni di basso livello, nelle quali si considereranno forme, dimensioni, colori, posizioni e traiettorie, ad altre di livello ben più alto, che riescono a fornire contenuto "semantico" al media, come la definizione dei soggetti presenti e le loro interazioni (es.: "scena nella quale vi è un cane che abbaia sulla sinistra, una palla blu che sta rimbalzando sulla destra, con il rumore di sottofondo di un'auto"). Ovviamente tra questi due casi estremi vi è un ampio spettro livelli intermedi di descrizione. Il fattore per il quale non è sempre consigliabile una descrizione di alto livello è che più la descrizione è "complessa" (nel significato di contenuto "semantico") più è difficile un'estrazione automatica dello stesso. E' ovvio infatti che, mentre il colore dominante di una scena è estraibile in modo automatico da applicativi dedicati, l'estrazione di "features" di più alto livello è ottenibile solo mediante un sempre maggiore intervento umano.

Cominciando quindi ad interessarci al modo con cui è opportuno descrivere un documento, è ovvio che in un'ottica di gestione di documenti multimediali, è necessario fornire innanzitutto informazioni generali, come:

- Tipo di memorizzazione utilizzata(Jpeg, mpeg, gif...);
- "size";
- condizioni e restrizioni per l'accesso;
- proprietario;
- autore;
- classificazione (p.e. rating del contenuto per la tutela dei minori);
- link ad altri materiali rilevanti;
- contesto (p.e. immagine catturata durante le olimpiadi del 1996, finale 200m uomini);
- ...

Oltre a queste informazioni di carattere generale l' MPEG-7 offre strumenti per la descrizione del contenuto del media, che possono includere, tra le altre, informazioni inerenti:

- la creazione e produzione (p.e. direttore, titolo...);
- l'uso (p.e. informazioni di copyright, "log", pianificazione);
- la struttura spaziale o spazio-temporale dei vari componenti (p.e. segmentazione di un filmato in scene);
- features di basso livello: colore dominante, texture, durata di un "file" sonoro...;
- features concettuali, di alto livello: soggetti, luoghi, azioni compiute...;
- ...

Ovviamente non è detto che la descrizione MPEG-7 sia presente "insieme" (nello stesso "stream dati") al documento d'origine; può tranquillamente essere dislocata altrove, né nello stesso computer ma neppure sulla stessa rete. In tal caso però sono necessari meccanismi bidirezionali di "sincronizzazione" tra la descrizione e il media d'origine.

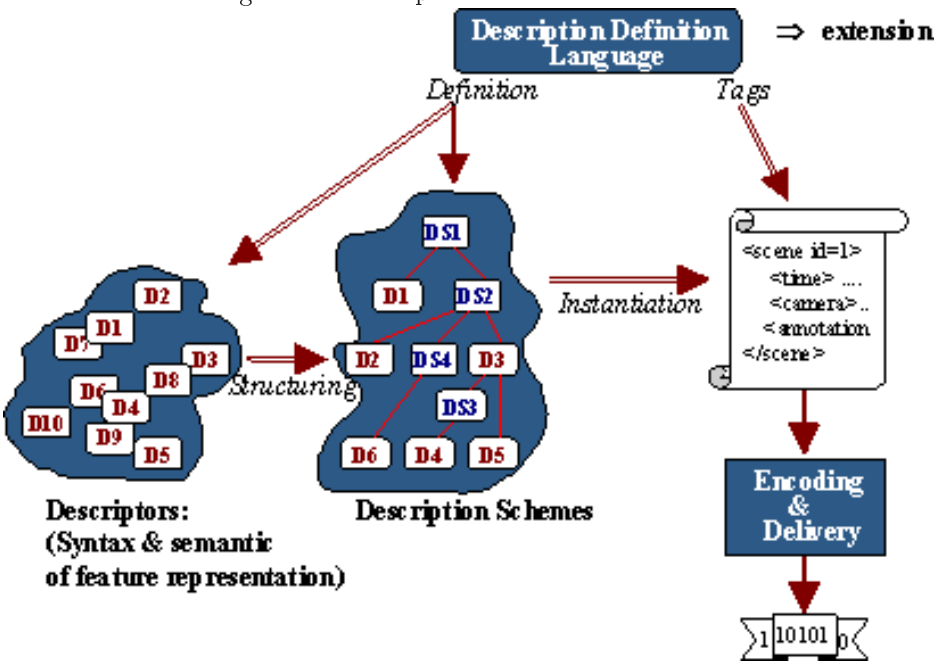
In sede di standard decisione fondamentale è stata quella di stabilire che le descrizioni dei documenti (N.B.: la loro forma "canonica") avvenga mediante documenti XML. Tali descrizioni però devono essere "aderenti" allo standard, ossia rispettare vincoli sulle descrizioni utilizzate, sulla loro struttura e cardinalità. Per fare ciò l'MPEG-7 standardizza (scopo fondamentale di questo standard) i cosiddetti "Descriptors" (D) e "Description Schemes" (DS), che non sono altro che le varie feature (le rappresentazioni di esse) che si possono utilizzare nelle generazione delle descrizioni XML. Stabilito che l' MPEG-7 utilizza D e DS per descrivere ogni tipo di media, si deve affrontare la modalità con le quali

questi elementi vengono codificati. Poiché tale standard tende ad essere il più generico possibile, si è scelto di utilizzare come forma di codifica testuale "canonica" il linguaggio "XML-Schema". Considerando infatti la popolarità sempre crescente dell'universo XML, il suo uso si pensa faciliterà l'interoperabilità tra le varie implementazioni di tale standard nel futuro.

Tocchiamo quindi il punto fondamentale dello standard, e cioè, come prima accennato, gli elementi caratteristici di esso, che risultano essere:

- "Descriptors" (D): rappresentazione di "features", descrivendole in termini di sintassi e semantica, descrivendo la relazione feature–>valore;
- "Description Schemes" (DS): rappresentazione della struttura e della semantica delle sue componenti, che possono essere sia D che DS; i vari DS sono, in poche parole, modelli dell'oggetto multimediale a cui si riferiscono e dell' "universo" che rappresentano, specificando il tipo dei D e DS che possono essere usati e le relazioni fra di essi;
- "Description Definition Language" (DDL): linguaggio che permette la creazione di nuovi DS e, possibilmente, D oltre all'estensione di DS già esistenti nello standard;
- Strumenti di sistema per il supporto di meccanismi come la sincronizzazione della descrizione con il media, meccanismi di trasmissione, rappresentazione codificata in forma testuale o binaria...

Figura 12: Principali elementi dell'MPEG-7



In figura 12 si possono notare le relazione fra questi elementi:
I D e DS (quelli standardizzati) sono divisi, al fine di descrivere correttamente tutte le informazioni suddette, in 5 grandi ambiti:

- Creazione & Produzione;
- Media;
- Uso;
- Aspetti strutturali;
- Aspetti concettuali.

Questi 5 differenti aspetti di una descrizione, qui presentati come entità separate, non sono in realtà totalmente disgiunti, ma sono correlati fra loro, e in alcune parti, anche inclusi l'un l'altro.

Come potete capire, MPEG-7 è uno standard che definisce delle regole per un tipo di annotazione riguardante strettamente il file video/audio in esame. Annotazione che serve principalmente per consentire un reperimento rapido ed efficiente di documenti video ed immagini, basato sul loro contenuto visuale e semantico. Adottare questo standard per il mio progetto non avrebbe portato a nulla. MPEG-7 non prevede infatti la scrittura di commenti, note personali da parte di chiunque lo desideri. Per non parlare poi delle ricerche e della modifica di questi commenti.

5 Conclusioni

Per poter riflettere sul lavoro svolto cerco di immedesimarmi in uno studente qualsiasi che vuole seguire una lezione universitaria a casa.

Prima del mio intervento si poteva trovare in una situazione di caos dove, alcune lezioni erano in ePresence, altre in Lode, scaricabili da due portali internet diversi. Le lezioni in ePresence inoltre erano la maggior parte e lo studente si trovava a dover seguire la lezione da casa privo di funzionalità molto utili presenti solamente in Lode.

in questo sistema ora ci sono tutte le lezioni, questo permette all'utente di connettersi ad un sito internet e di trovare tutti i corsi digitalizzati nel formato più efficiente.

Lode, come detto, ha due versioni. Il mio lavoro consente di avere ogni lezione in entrambe i formati (Javascript+Java e Flash) per offrire all'utente persino una scelta su come seguire il corso desiderato.

Un altro aspetto molto importante riguarda il responsabile del progetto Lode: è per lui indispensabile avere conoscenza piena di tutte le informazioni che ha a disposizione su ogni lezione. Prima della conversione questo non era necessario in quanto i corsi non erano così tanti. Ora questa affermazione non è più valida ed il database da me costruito serve proprio a questo. Con un'appropriata query di ricerca è possibile trovare tutto quello che si desidera su ogni singola slide, corso, lezione o professore che la tiene.

La versione di Lode in Flash apre le porte ad una nuova funzionalità che in futuro potrebbe diventare molto utile ed apprezzata dagli utenti che sfruttano il sitema: l'annotazione dei video. Prima di tutto ciò non era possibile per nessuno interagire, scrivere opinioni, giudizi riguardanti una specifica lezione o slide. Il lavoro da me descritto nel capitolo 4 serve proprio a questo scopo. In futuro il sito internet potrebbe essere messo on-line ed esprimere un pensiero su una lezione condividendolo con chiunque fosse interessato, non sarebbe più impossibile. Riprendendo inoltre l'osservazione fatta nel paragrafo 4.2.3, si potrebbe più avanti pensare ad un collegamento tra i due database per permettere a chiunque di ricercare e visualizzare tutte informazioni che si possono desiderare su qualsiasi corso.

Un'ultima considerazione la dedico all'esperienza fatta in questi mesi di studio e progettazione. Veder nascere un progetto, dedicare del tempo per una cosa che può tornar utile a molte persone è sicuramente stimolante, le giornate passano senza che te ne possa accorgere e lavorare diventa un piacere. Tutto ciò mi ha fatto crescere e mi ha permesso di imparare nuovi linguaggi, strumenti software, ma anche tanti aspetti che sono socuro mi serviranno in futuro.

6 Bibliografia

Riferimenti bibliografici

[1] Dolzani M., Ronchetti M.,
"Video Streaming over the Internet to support learning: the LODE system" WIT Transactions on Informatics and Communication Technologies, 2005, v. 34, p. 61-65.

[2] Dolzani M., Ronchetti M. (2005).
Lectures On DEmand: the architecture of a system for delivering traditional lectures over the Web. In Kommers, P., & Richards, G. (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2005 (pp. 1702-1709). Chesapeake, VA: AACE.

[3] Ronchetti M.,
"Has the time come for using video-based lectures over the Internet? A Test-case report" "CATE - Web Based Education Conference 2003", Rhodes (Greece), June 30 - July 2, 2003

[4] Ronchetti, M. (2003).
Using the Web for diffusing multimedia lectures: a case study. In Kommers, P., & Richards, G. (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003 (pp. 337-340). Chesapeake, VA: AACE.

[5] Guida avanzata di scripting Bash,
PLUTO Project, http://www.pluto.it/files/ildp/guide/abs/index.html

[6] Guida CSS di base, CSShtml.it,
http://css.html.it/guide/leggi/2/guida-css-di-base/

[7] Guida Layout dei siti con i CSS, CSShtml.it,
http://css.html.it/guide/leggi/3/guida-layout-dei-siti-con-i-css/

[8] Guida dei tag form, HTML/XHTMLhtml.it,
http://xhtml.html.it/guide/lezione/1696/struttura-del-tag-form/

[9] Guida PHP su Linux, PHPhtml.it,
http://php.html.it/guide/leggi/92/guida-php-su-linux/

[10] PHP Tutorial,
http://www.w3schools.com/php/

[11] Guida alle sessioni in PHP, SIForce.org,
http://www.siforge.org/articles/2003/11/10-guida_sessioni_php.html

[12] Guida PHP/MySql pratica, PHPhtml.it,
http://php.html.it/guide/leggi/77/guida-phpmysql-pratica/

[13] Video Annotation Tool, Project Pad
http://dewey.at.northwestern.edu/ppad2/documents/help/video.html

[14] Diver, Diogital Interactive Video Exploration & Reflection
http://diver.stanford.edu/home.html

[15] ANVIL (ANnnotation of VIdeo and Language data)
http://www.anvil-software.de/documentation.html

[16] MPEG-7, Giuseppe Raffa
http://digilander.libero.it/beppewww/mpeg/

[17] MPEG-7 Overview, José M. Martínez
http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm