POLITECNICO DI MILANO
Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

lezi.NET
Sistema di erogazione della didattica

Relatore: Prof. **Paolo Paolini**
Correlatore: Dott. **Luca Mainetti**

Tesi di Laurea di:
Bruna Stefano
Matricola 623670

Anno Accademico 2002-2003

a mia madre e mio padre

Ringraziamenti

Ringrazio i miei genitori per avermi dato la possibilità di raggiungere questo per me importantissimo e sempre sognato traguardo. Un grazie a Sabrina che ha sacrificato momenti del nostro tempo libero per permettermi di studiare e lavorare. Un grazie ad Alexio per avermi sopportato e supportato in questi duri anni di lavoro e studio. Una grazie a tutto il laboratorio HOC e alla "fattoria elettronica" per aver reso piacevoli, con la loro simpatia, le giornate di lavoro.
E infine un grande grazie a tutti coloro che mi hanno aiutato nella realizzazione di questo lungo progetto, in particolare il Prof. Paolo Paolini per la sua fiducia e il Dott. Luca Mainetti per le grandi opportunità che mi ha dato e la sua enorme disponibilità.

Stefano.

INDICE

1 INTRODUZIONE ..... - 1 -

Parte 1 : Background

2 E-LEARNING ..... - 4 -

2.1 Istruzione a distanza ..... - 4 -
2.1.1 Learning management system ..... - 7 -

2.2 Progetto virtual campus ..... - 11 -

3 STANDARD ..... - 13 -

3.1 IMS ..... - 13 -
3.1.1 Specifiche prodotte ..... - 14 -
3.1.2 Content Packaging specification ..... - 16 -
3.1.3 Struttura del manifest ..... - 17 -

3.2 ADL ..... - 21 -
3.2.1 SCORM 1.2 ..... - 21 -
3.2.2 Content aggregation model ..... - 22 -
3.2.3 Run-Time environment ..... - 24 -
3.2.4 Interfaccia di comunicazione (API) ..... - 24 -
3.2.5 Data model ..... - 26 -

4 TECNOLOGIE ..... - 32 -

4.1 SOAP - WSDL– UDDI ..... - 32 -

4.2 Microsoft .NET Framework ..... - 35 -
4.2.1 Common language runtime e librerie di classi. ..... - 35 -
4.2.2 Common type system ..... - 40 -

4.3 C# ..... - 41 -

4.4 Pagine Web server-side ASP .NET ..... - 42 -

Parte 2 : Lavoro svolto

5 REQUISITI ..... - 46 -

6 PROGETTAZIONE ..... - 47 -

6.1 Ruoli, gruppi e utenti ..... - 47 -

6.2 Distribuzione dei diritti: ACL e policies ..... - 47 -

6.3 Funzionalità ..... - 49 -

6.4 Classi e strutture dati ..... - 65 -
6.4.1 Strutture dati per i modelli ..... - 68 -

7 ARCHITETTURA ..... - 75 -

7.1 Struttura funzionale generale ..... - 75 -

7.2 Struttura fisica generale ..... - 77 -

7.3 Struttura della pagina ..... - 80 -

7.4 Interfaccia utente ..... - 82 -

8 IMPLEMENTAZIONE ..... - 85 -

8.1 Applicazione Web Lezi.NET ..... - 85 -
8.1.1 Lezi.NET Viewer ..... - 87 -
8.1.2 Lezi.NET Resource manager ..... - 93 -
8.1.3 Lezi.NET Editor ..... - 95 -
8.1.4 Lezi.NET Administrator ..... - 102 -
8.1.5 Lezi.NET streaming server ..... - 103 -
8.1.6 Proxy ..... - 104 -

8.2 Applicazione Web Lezi.NET content management system ..... - 105 -
8.2.1 Modulo di autenticazione ..... - 107 -
8.2.2 Modulo di autorizzazione ..... - 108 -
8.2.3 Moduli di gestione delle entità ..... - 109 -

8.3 Base di dati ..... - 109 -
8.3.1 Schema ..... - 109 -
8.3.2 Stored procedure ..... - 112 -

9 PRESTAZIONI ..... - 115 -

10 PIATTAFORMA DI SVILUPPO E PIATTAFORMA OPERATIVA ..... - 119 -

11 VALIDAZIONE E SPERIMENTAZIONE ..... - 124 -

11.1 Acquisizione e produzione ..... - 124 -

11.2 Impressioni d'uso ..... - 128 -

12 CONCLUSIONI ..... - 130 -

BIBILIOGRAFIA E RIFERIMENTI ..... -132-

Appendice A

MANUALE UTENTE ..... -135-

Login ..... -135-

Start ..... -136-

Profilo ..... -138-

Corsi ..... -139-

Corso ..... **-140-**

Appendice B

MANUALE UTENTE CREATORE ..... **-143-**

Groups ..... **-143-**

Users ..... **-144-**

News ..... **-147-**

Subscriptions ..... **-148-**

Projects ..... **-149-**

Resources ..... **-152-**

Corso (estensione) ..... **-154-**

Appendice C

MANUALE UTENTE AMMINISTRATORE ..... **-156-**

Installazione applicazione ..... **-156-**

1 Introduzione

Il progetto lezi.NET nasce con l'intento di creare un ambiente di produzione e fruizione di contenuti didattici pubblicabili in rete.
Gli obiettivi di questo progetto sono quindi incentrati sulla progettazione e seguente implementazione di un sistema di erogazione della didattica, o learning management system, che permetta di soddisfare i requisiti fondamentali brevemente descritti di seguito.

- **Ambiente di fruizione Web-based**: l'utente tramite il proprio browser deve poter fruire i contenuti di un corso in modo semplice ed automatico mediante un'interfaccia coerente e usabile.
- **Ambiente di gestione dei corsi Web-based**: gli utenti della piattaforma, in base ai loro diritti, devono avere una visione personalizzata della lista dei corsi presenti nel sistema.
- **Ambiente di editing dei corsi Web-based**: gli utenti appartenenti al gruppo di "creatori di corsi" devono poter creare lezioni utilizzando delle risorse in precedenza archiviate nel sistema o caricabili anche durante la fase di produzione.
- **Ambiente di gestione delle risorse Web-based**: deve essere possibile caricare nel sistema le risorse (filmati, diapositive, documenti etc) che potranno essere utilizzate in fase di editing dei corsi in modo da avere un archivio personale, eventualmente condiviso, del proprio materiale didattico.
- **Ambiente di gestione delle iscrizioni ai corsi Web-based**: per non obbligare l'utente creatore di corsi a distribuire manualmente i diritti agli utenti fruitori per un suo corso, deve essere disponibile una procedura guidata che faciliti tale compito.
- **Ambiente di gestione degli utenti e dei gruppi Web-based**: deve essere disponibile uno strumento che permetta gestire utenti e gruppi di utenti per poter supportare processi di autenticazione e autorizzazione.

Le motivazioni che hanno portato al progetto lezi.NET riguardano principalmente la volontà di realizzare e verificare l'efficacia di un sistema che permetta, oltre che l'erogazione di contenuti, anche la loro produzione. Un sistema di questo tipo deve essere in grado di velocizzare e semplificare l'intero ciclo didattico che va appunto dalla fase di acquisizione delle risorse a quella di fruizione dei contenuti. Il risultato di questa operazione dovrebbe permettere di ottenere un migliore rapporto fra efficacia e costi della distribuzione di contenuti didattici. I continui progressi tecnologici nel dominio di Internet e le caratteristiche proprie della rete forniscono un valido supporto tecnologico per il soddisfacimento dei requisiti e l'implementazione della piattaforma, che quindi è stata pensata e realizzata come applicazione Web.
Un altro obiettivo di cui si è tenuto conto durante la realizzazione del progetto è stato quello di permettere di utilizzare il componente di fruizione lezi.NETViewer (conforme allo standard SCORM 1.2[1]) all'interno di una piattaforma di learning più articolata come quella di **Virtual Campus**[2] **(VC)**. In questo modo lezi.NET, oltre ad essere un semplice learning management system autosufficiente, può divenire parte integrante di una

[1] *Vedi Sezione 3.2.1*
[2] *Vedi sezione 2.2*

architettura più estesa funzionando come ambiente di visualizzazione di contenuti didattici (Learning Objects). In questo modo il motore di workflow della piattaforma VC che regola i cammini didattici di attraversamento dei LOs è in grado di sfruttare il lezi.NETViewer come ambiente di fruizione per learning objects atomici. VC è infatti in grado di pilotare l'esecuzione LO atomici facenti parte di LOs complessi su diverse piattaforme di esecuzione come il leziNETViewer. Per tale motivo le maggiori attenzioni e sforzi implementativi sono stati dedicati alla realizzazione dell'ambiente di fruizione con particolare riguardo al soddisfacimento dei requisiti definiti dallo standard SCORM.

Le diverse fasi del processo di sviluppo, analisi dei requisiti, progettazione, implementazione, verifica e validazione sono descritte nel presente documento. Questo progetto ha, tuttavia, richiesto due diverse attività di ricerca precedenti il normale ciclo di produzione. La prima, di tipo tecnologico, dovuta alla necessità di conoscere le attuali tecnologie e piattaforme per l'implementazione di sistemi distribuiti. La seconda riguardante, invece, il dominio dell'applicazione, l'e-learning, per avere una visione chiara dei sistemi esistenti e degli standard disponibili. Un sunto di queste attività di scouting è presente nella Parte Background del documento costituita dalle seguenti sezioni principali:

- Sezione 2: Analisi dei concetti principali e delle diverse entità coinvolte nel dominio dell'e-learning.
- Sezione 3: Descrizione dei principali standard esistenti utilizzati in seguito per la realizzazione del progetto.
- Sezione 4: Descrizione delle principali tecnologie utilizzate durante la fase di implementazione del sistema.

La seconda parte del documento riguarda, invece, la descrizione del lavoro svolto. Sono di seguito elencate le sezioni con una breve descrizione introduttiva.

- Sezione 5 - Requisiti: La sezione descrive brevemente i principali requisiti che l'applicazione realizzata deve essere in grado di soddisfare.
- Sezione 6 – Progettazione : Descrizione della fase di progettazione. Il livello di dettaglio raggiunto in questa parte del documento varia a seconda dell'argomento. Tale sezione non pretende di essere un documento di progetto completo ma solo un "estratto" di una documentazione di progetto che sarebbe stata sicuramente più ampia e, soprattutto, pensate da leggere a causa del frequente utilizzo di notazione semiformale come l'UML.
- Sezione 7 - Architettura: Alla descrizione dell'architettura funzionale e fisica del sistema, pur facendo della progettazione, è stata dedicata una sezione a parte vista la complessità che può raggiungere un sistema di learning, soprattutto in ambiente distribuito. In questa sezione si descrive anche l'architettura di componenti più semplici che sono utilizzati spesso nell'applicazione.
- Sezione 8 – Implementazione: La descrizione dell'attività di implementazione spiega, più o meno dettagliatamente, le soluzioni che sono state utilizzate per la realizzazione delle funzionalità necessarie per la realizzazione dell'intero sistema. Anche in questo caso non è stato possibile parlare di tutto ma si è cercato di riportare solo le soluzioni principali o più originali.
- Sezione 9 – Prestazioni: La realizzazione di una breve analisi delle prestazioni è stata suggerita da alcune semplici considerazioni. Essendo l'architettura discretamente complessa (come si vedrà in seguito lezi.NET è un'applicazione Web multi tier), utilizzando una nuova piattaforma di sviluppo e tecnologie

relativamente recenti si è voluto testare l'efficienza e le prestazioni di una installazione semplice dell'applicazione.

- Sezione 10 – Piattaforma di sviluppo e piattaforma operativa. Come noto, una della fasi più critiche dell'intera vita di un software è la sua installazione è messa in opera. Si è dedicato quindi una breve sezione alla descrizione delle diverse piattaforme hardware e software sulle quali l'applicazione è stata installata ed, in certi, casi anche testata.
- Sezione 11 – Esempio: Corso di esempio atto a dimostrare le funzionalità e l'efficacia del sistema nelle sue diverse modalità operative.
- Sezione 12 – Conclusioni: Considerazioni finali dopo un certo periodo di operatività del sistema.

Nella parte finale del documento sono presenti tre appendici che insieme costituiscono un breve manuale utente dell'applicazione.

- Appendice A - Manuale utente: Manuale dell'utente che accede al sistema per fruire i contenuti pubblicati.
- Appendice B - Manuale utente creatore : Manuale dell'utente che accede al sistema per creare contenuti successivamente fruibili.
- Appendice C - Manuale utente amministratore: Manuale di deployment, installazione e manutenzione del sistema.

Parte 1 : Background

2 e-Learning

2.1 Istruzione a distanza

Una definizione d'istruzione a distanza, anche se poco esaustiva, potrebbe essere quella di "metodo d'apprendimento esterno alla classe scolastica". Tale definizione sottintende l'istruzione a distanza come semplice autoistruzione non tenendo conto del fatto che ogni individuo possa utilizzare un metodo d'apprendimento personale. Una visione non limitata al semplice studio individuale dell'istruzione a distanza include un'organizzazione dell'autoistruzione personalizzata, l'utilizzo di materiale fornito dall'educatore ed un insieme di "servizi", come la possibilità di contattare direttamente il docente, finalizzati al successo dello studente.
Utilizzando questi concetti nascono le prime forme di IAD in cui viene a mancare, per gran parte del processo d'apprendimento, il contatto fra discente e docente. I contenuti da trasmettere allo studente utilizzano la tecnologia disponibile del tempo: materiale cartaceo spedito tramite posta tradizionale.
Per quanto riguarda il canale di comunicazione inverso, cioè quello che parte dallo studente e arriva al docente, sono utilizzati esercizi scritti spediti al docente che dopo averli corretti interagisce ulteriormente fornendo una valutazione, una correzione ed eventualmente dei suggerimenti.
Ciò che differenzia l'istruzione a distanza da un pacchetto per l'autoistruzione è proprio la presenza di un'organizzazione capace di condurre e monitorare l'andamento dell'apprendimento dello studente.
Un approccio di questo tipo è in grado di conservare i vantaggi dell'istruzione tradizionale come l'interazione fra docente e studente, aggiungendo i vantaggi della IAD come la possibilità da parte dello studente di organizzare i tempi e, limitatamente, i modi del proprio processo d'apprendimento.
Le differenze principali presenti fra istruzione tradizionale e quella a distanza sono quindi i diversi canali di comunicazione: voce con contatto "faccia a faccia" fra studente e docente e fra studenti contro trasmissione bidirezionale di informazioni in modo sincrono o asincrono dipendente tuttavia dalla tecnologia disponibile.
Mediante il miglioramento tecnologico si cerca di portare all'efficienza e naturalezza della divulgazione verbale i canali di comunicazione della IAD. L'avvento delle nuove tecnologie, in primo luogo di internet, ha dato la possibilità di rendere l'apprendimento a distanza maggiormente interattivo e dinamico facendo nascere l'e-Learning.

La formazione in rete ha dato la possibilità di dar vita ad un sistema che coniughi con successo, l'istruzione a distanza classica e l'istruzione in presenza, integrando le caratteristiche fisiche della prima con quelle psicologiche della seconda. L'abbattimento delle barriere spazio-temporali dovuto all'apprendimento elettronico comporta un insieme consistente di vantaggi sia per lo studente che per il docente:

- **Facilità di raggiungimento delle risorse**: qualsiasi tipo di contenuto che sia statico od interattivo può essere memorizzato in formato elettronico e quindi essere distribuito facilmente in rete. E' necessaria quindi solo una connessione ad Internet per poter avere un punto d'accesso ai contenuti.

- **Abbattimento dei costi**: da parte di chi produce i contenuti viene a mancare la necessità di trovare luoghi adatti per l'insegnamento quali aule e sale e da parte dello studente non è più necessario recarsi fisicamente presso tali luoghi con grande risparmio di tempo e denaro.
- **Grado di personalizzazione**: condividendo, durante una lezione classica, lo stesso ambiente, è, utilizzata una singola metodologia di insegnamento, imponendo omogeneità in un gruppo eterogeneo di studenti. L'e-Learning può permettere un alto grado di personalizzazione dei contenuti in modo che ciascuno studente possa scegliere quello che più si adatta al proprio meccanismo d'apprendimento. In un'aula i tempi d'esecuzione sono rigidi e non tengono conto delle differenti esigenze dei membri del gruppo. Inoltre, in molti contesti, durante una lezione in aula l'applicazione operativa dei concetti appresi può non essere realizzabile.
- **Monitoraggio del livello di apprendimento:** mediante un sistema di e-Learning è possibile seguire l'andamento dell'apprendimento in modo continuo ed economico magari mediante piccole verifiche poste in punti chiave del percorso didattico. Tali verifiche, come per le normali lezioni, non comportano la necessità di trovare luoghi adatti in cui svolgerle.
- **Carico di responsabilità**: una continua osservazione del livello d'acquisizione permette anche di responsabilizzare lo studente, con maggiore continuità, durante tutta la fase d'apprendimento invece di alternare lunghi periodi didattici a pesanti verifiche consuntive.
- **Contenuti al centro dell'attenzione**: l'e-Learning permette di emanciparsi dall'obbligo della condivisione spazio-temporale rendendo l'educatore remoto o, addirittura, simbolicamente esistente solo nella mente dell'allievo permettendo di centralizzare l'attenzione esclusivamente sull'oggetto di studio.
- **Vantaggi sociali**: tutti devono avere l'opportunità di apprendere, anche coloro che sono impossibilitati a muoversi, geograficamente isolati o socialmente svantaggiati. L' e-Learning è quindi un'opportunità che può potenzialmente coinvolgere un maggior numero di persone includendo quelle impossibilitate dalle caratteristiche dell'istruzione classica.

Sono individuabili tuttavia dei potenziali problemi nel modello dell'e-Learning. Il problema più evidente nell'IAD classica è la scarsa qualità del canale di comunicazione per l'interazione fra studente e docente. Si ha in pratica il dubbio che con i mezzi che la tecnologia mette a disposizione non si riesca a raggiungere il livello comunicativo della divulgazione verbale in presenza. Tuttavia, soprattutto mediante internet, oggi si possono utilizzare diversi canali di comunicazione per soddisfare le più disparate necessità. Ad esempio, per una semplice domanda è possibile spedire al docente un e-Mail oppure riempire una form di un sito di "assistenza didattica", invece per una discussione che verte su concetti "difficili" o "costosi" da scrivere, è possibile utilizzare la videoconferenza, esperienza molto vicina al dialogo diretto.
Un'altra critica che si potrebbe fare sul modello dell'IAD è la mancanza oltre che del rapporto diretto studente-docente anche del rapporto diretto fra studenti. L'importanza di tale rapporto durante l'educazione, e soprattutto, la maturazione sociale dello studente è evidente. Nessun sistema d'apprendimento a distanza sarà mai in grado di sostituire le esperienze di vita maturate durante le lezioni in classe. Tuttavia, l'e-Learning non si pone come elemento sostitutivo della normale interazione sociale ma come sistema d'apprendimento interattivo che non preclude i momenti d'incontro e socializzazione necessari per la crescita globale di un individuo. Ci possono essere, tuttavia, casi in cui

anche in una classe tradizionale esistono concetti di distanza diversi da quella fisica. Può, ad esempio, esistere un distanza sociale causata da idee e pregiudizi di alcuni membri del gruppo che potrebbe rendere difficile la convivenza in classe. Oppure, più semplicemente, in un gruppo di grandi dimensioni il dialogo e l'interazione possono diminuire mentre con l'e-Learning, mediante, ad esempio un forum, è possibile garantire pari opportunità di interazione a tutti i partecipanti.

Nonostante le evidenti buone caratteristiche dell'e-Learning e l'ottimo supporto tecnologico attuale, è ancora diffusa la convinzione che l'educazione, per essere riconosciuta in quanto tale, si debba svolgere attraverso un rapporto di compresenza spazio-temporale tra l'allievo e l'educatore. Un pregiudizio di questo tipo è dovuto a motivazioni sociali e culturali. L'informazione per diversi secoli è stata tramandata oralmente e proprio la comunicazione diretta è stato il primo strumento didattico. Si riteneva che il rapporto educativo dovesse aver luogo necessariamente in presenza e si riteneva che l'educazione dovesse essere fondata su scambi diretti fra docente e discente. Quando arrivò la scrittura nel V secolo a.C., furono proprio i maestri a non accettare questa nuova forma educativa considerandola inadattabile alle necessità sia degli allievi che dei docenti. Venti secoli dopo la storia si ripeteva, e anche la stampa, perlomeno la sua valenza educativa, fu denigrata a tal punto che le università del tempo ne vietavano l'uso per scopi didattici. Si deve arrivare fino al XVIII e XIX per vedere come scrittura e stampa, vere e proprie rivoluzioni tecnologiche e culturali, costituiscano il fondamento di base della scuola e di tutti i processi formativi. L'avvento dei mass media, come radio e televisione, ha permesso di accedere ad una quantità di informazioni enorme. Nonostante ciò, l'apprendimento rimane ancora di tipo recettivo divulgato da un unico centro di distribuzione. Solo negli anni '90, momento in cui nasce la comunicazione mediata da computer, sono realmente presenti le potenzialità per lo sviluppo di un modello di conoscenza in rete attivo e partecipato. Probabilmente oggi sta accadendo la stessa cosa: le potenzialità dell'e-Learning non vengono ancora completamente comprese ma potrebbe succedere che in futuro, come accaduto in passato per la stampa, l'e-Learning diventi la metodologia standard per l'apprendimento.
La speranza è quella che l'implementazione di sistemi di e-Learning, mediante le nuove tecnologie, permetta di superare i pregiudizi e gli svantaggi pratici dei precedenti modelli di apprendimento a distanza.

2.1.1 Learning management system

Il termine "Learning management system", di solito indicato con la sigla LMS, indica un insieme di funzionalità principali progettate per distribuire e gestire contenuti, interazioni e progressi degli studenti.
Il termine LMS può essere applicato sia a piccoli sistemi locali che permettono, ad esempio, la fruizione di un corso su un CD-ROM, che a sistemi complessi distribuiti in cui è inclusa anche la gestione dell'utenza, processi d'autorizzazione, autenticazione, workflow etc.
Un modello generale che mostra i componenti e i servizi potenzialmente presenti in un'implementazione di un LMS è visibile in Figura 1.

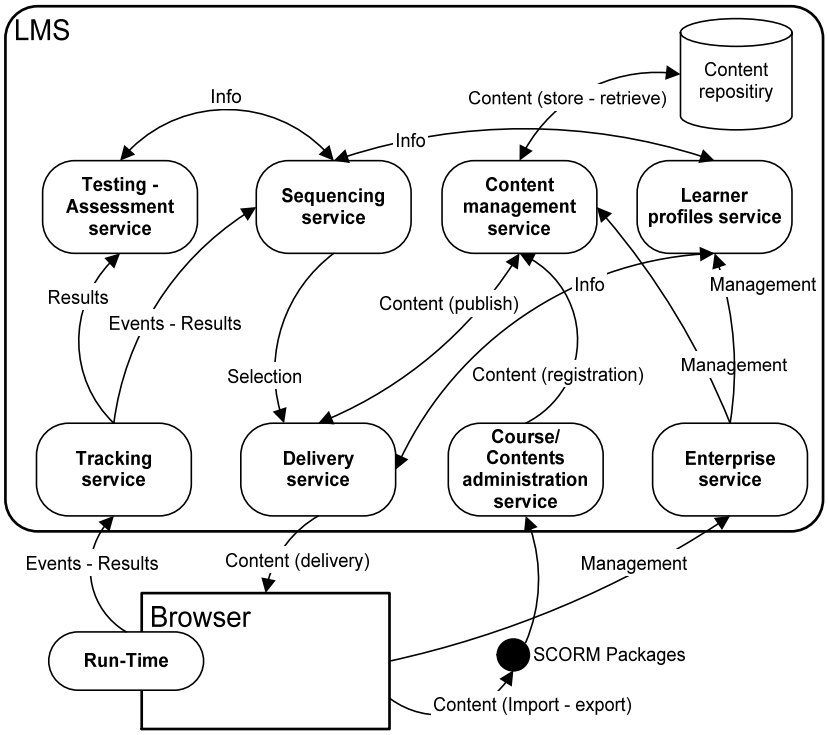


Figura 1: componenti e servizi di un LMS generico.

E' presentata di seguito una breve descrizione dei componenti del LMS generico di Figura 1. Tuttavia l'architettura proposta non è da considerarsi rigida in quanto alcune funzionalità possono essere portate da un componente all'altro dipendentemente dal punto di vista e dai dettagli implementativi che inevitabilmente possono condizionare la progettazione. Inoltre alcuni servizi possono essere riuniti in un unico modulo funzionale.

- **Run-Time:** è l'ambiente in cui il contenuto viene eseguito. Permette l'interazione fra componente di presentazione, generalmente un browser, e LMS quando la comunicazione è iniziativa del contenuto. I contenuti eseguibili in un Run-Time possono comunicare attivamente, tramite logica di controllo contenuta negli stessi[3], con il sistema. L'ambiente Run-Time fornirà quindi delle

[3] *Tale logica si occupa principalmente della gestione dello stato del contenuto, ad esempio se la fruizione è stata completata correttamente, e della lettura/salvataggio di informazioni relative all'istanza di esecuzione corrente identificata dal contenuto stesso e dall'utente. Generalmente il contenuto è basato su pagine HTML che contengono*

API che potranno essere usate per interagire con l'LMS. Ad esempio, una volta che il contenuto in esecuzione è terminato, tramite l'API del Run-Time è possibile comunicare tal evento di stato in modo che l'LMS sia in grado di determinare l'azione successiva come ad esempio la messa in esecuzione del contenuto successivo. Le API forniscono, generalmente, anche la possibilità di ottenere e/o salvare informazioni relative allo studente e allo stato della fruizione corrente come ad esempio tempi d'esecuzione, risultati di tests etc. Esistono diverse soluzioni implementative per realizzare tale componente: si può utilizzare un'applet client-side eseguita dal browser che si occupa di tradurre le richieste dei contenuti verso l'LMS oppure un componente JScript che, tramite Web Services, giri le chiamate API del contenuto al LMS.

- **Delivery service:** servizio che mette a disposizione dell'ambiente di esecuzione i contenuti. E' generalmente implementato mediante un Web server. I contenuti provengono dal Content management service. Le informazioni riguardanti quale contenuto visualizzare provengono dal Sequencing service.

- **Tracking service:** servizio che si occupa di gestire la comunicazione fra il contenuto in esecuzione e il resto del sistema. Fornisce l'implementazione, generalmente server-side, delle API che il contenuto può utilizzare nel Run-Time. Viene utilizzato il termine "Tracking" poiché una delle funzioni principali di questo modulo è quella di "tracciare" lo stato del contenuto (inizializzato, terminato, completato, errore etc) e di registrare informazioni relative la fruizione (risultato di un test, tempi di esecuzione etc). Ad esempio, un contenuto, per poter visualizzare il nome dello studente nella pagina, esegue una chiamata ad un metodo dell'API del Run-Time, quest'ultimo eseguirà una chiamata, ad esempio tramite Web Services, al Tracking service che troverà il nome dello studente in una struttura dati[4] preparata appositamente dal sistema per gestire l'istanza in esecuzione identificata dalla coppia contenuto-studente.

- **Sequencing service:** servizio che, in base allo stato del contenuto precedentemente visualizzato (ad esempio terminato correttamente/terminato non correttamente) e allo stato dell'utente (ad esempio possibilità di fruire o no di un determinato contenuto a causa di precedenze e/o autorizzazioni) decide quale sarà il prossimo contento da sottoporre all'utente. Tale servizio dovrebbe poter avere accesso alle informazioni relative all'utente, al contenuto e sempre alle informazioni relative all'utente però nel contesto di esecuzione. In Figura 1 è visibile una possibile interazione fra Browser, Sequencing service, Tracking service, Delivery service ed un ipotetico modulo che gestisce lo stato dell'istanza di esecuzione contenuto-utente.

- **Content management service:** servizio che pubblica sul Delivery service i contenuti che devono essere resi disponibili in base a quali corsi sono presenti nell'LMS. I contenuti sono prelevati da un repository ove sono stati in precedenza memorizzati ed organizzati.

- **Testing – Assessment service:** servizio che gestisce tutte le informazioni riguardanti i risultati e i progressi degli studenti.

codice JScript per gestire l'interazione con l'API del Run-Time. Nel capitolo 3.2.3 viene spiegato dettagliatamente il Run-Time SCORM.

[4] *La struttura dati nell'implementazione SCORM è il modello CMI. Vedi sezione 3.2.5*

- **Learner profile service:** servizio che gestisce le informazioni relative all'utente non relative ad alcun contesto di esecuzione. Ad esempio preferenze riguardanti la lingua dell'utente.

- **Enterprise service:** servizio per la gestione di utenti, gruppi, annidamento di gruppi, permessi, access control lists, autorizzazioni e autenticazione.

- **Content repository:** repository dei contenuti. Per eseguire operazioni di inserimento di contenuti è necessario definire un formato comune[5] (fra vari LMS) per garantire l'interscambio di contenuti o aggregazione di contenti come ad esempio corsi.

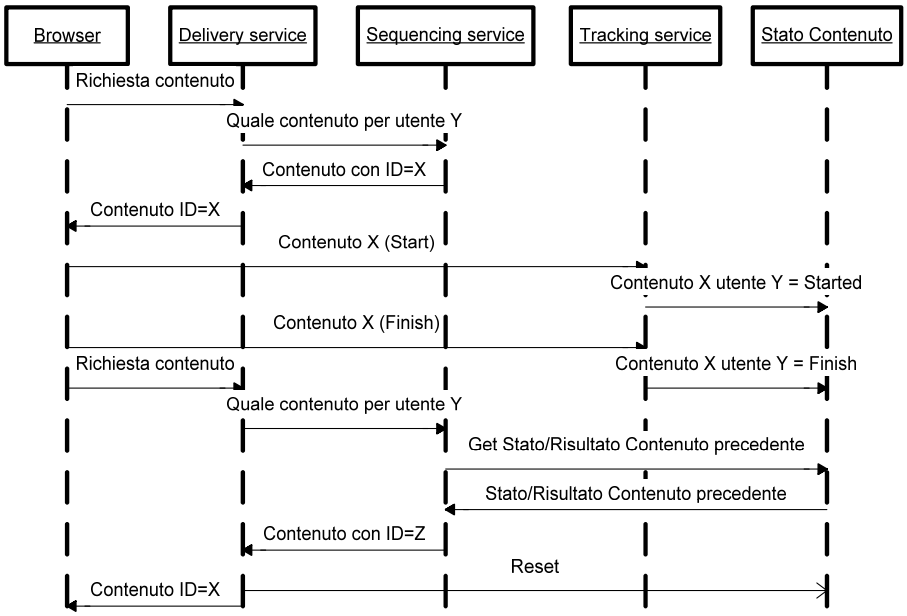


Figura 2: Sequence diagram che descrive la sequenza di interazioni fra alcuni componenti dell'LMS

L'architettura di Figura 2, pur essendo generica e flessibile è basata su alcune idee fondamentali.
In primo luogo l'alta modularità è stata utilizzata per facilitare un'implementazione del sistema in ambiente distribuito. In un'implementazione "estrema" è possibile tradurre ogni modulo funzionale nel corrispondente componente fisico. L'implementazione opposta prevede, invece, la realizzazione di un'unica applicazione Web pubblicata su Web server che identifica l'LMS nella sua interezza. Le soluzioni intermedie sono quelle che vengono più frequentemente utilizzate.
Una seconda idea base, anche se non completamente realizzabile con gli standard attuali, è quella che i contenuti dovrebbero essere organizzati in unità di piccole dimensioni[6] aggregando le quali è possibile costruire capitoli, moduli, corsi etc. Questo modo di operare

[5] *Un esempio di formato per l'interscambio di contenuti è la specifica di Content Packaging di IMS. Vedi sezione 3.1.2*
[6] *Negli standard presentati in seguito (SCORM - sezione 3.2 – 3.2.1) tali unità elementari di contenuto vengono chiamate SCO (sharable content object)*

permette di considerare le unità di contenuti come componenti elementari costituenti di aggregazioni più vaste, permettendo così un largo riutilizzo di unità e risorse. Ciò implica che le unità di contenuto debbano essere il più indipendenti possibile rispetto al contesto in cui vengono create e fra di loro, in modo da poterle riutilizzare anche in contesti diversi. Prese singolarmente le unità di contenuto non dovrebbero avere un contesto specifico. E' l'LMS, che scegliendo quali unità proporre, crea, più o meno dinamicamente, il corso visto come sequenza di learning objects. Per questo motivo, una regola fondamentale che non dovrebbe essere mai violata impone che una unità non possa "lanciare" un'altra unità di contenuto. Tale operazione è compito esclusivo dell'LMS. La navigazione all'interno della singola unità è naturalmente possibile.
Lo spostare il controllo del flusso d'esecuzione dai contenuti ad un LMS, permette oltre che il riutilizzo delle risorse anche la possibilità di realizzare grandi repository di contenuti e, dopo aver definito degli standards di interscambio, di formare un sorta di "content economy" in cui gli oggetti di scambio sono le unità di contenuto.

2.2 Progetto virtual campus

Il progetto Virtual Campus del Politecnico di Milano è una piattaforma per la progettazione, la fruizione e la valutazione di contenuti didattici . I suoi obiettivi principali sono :

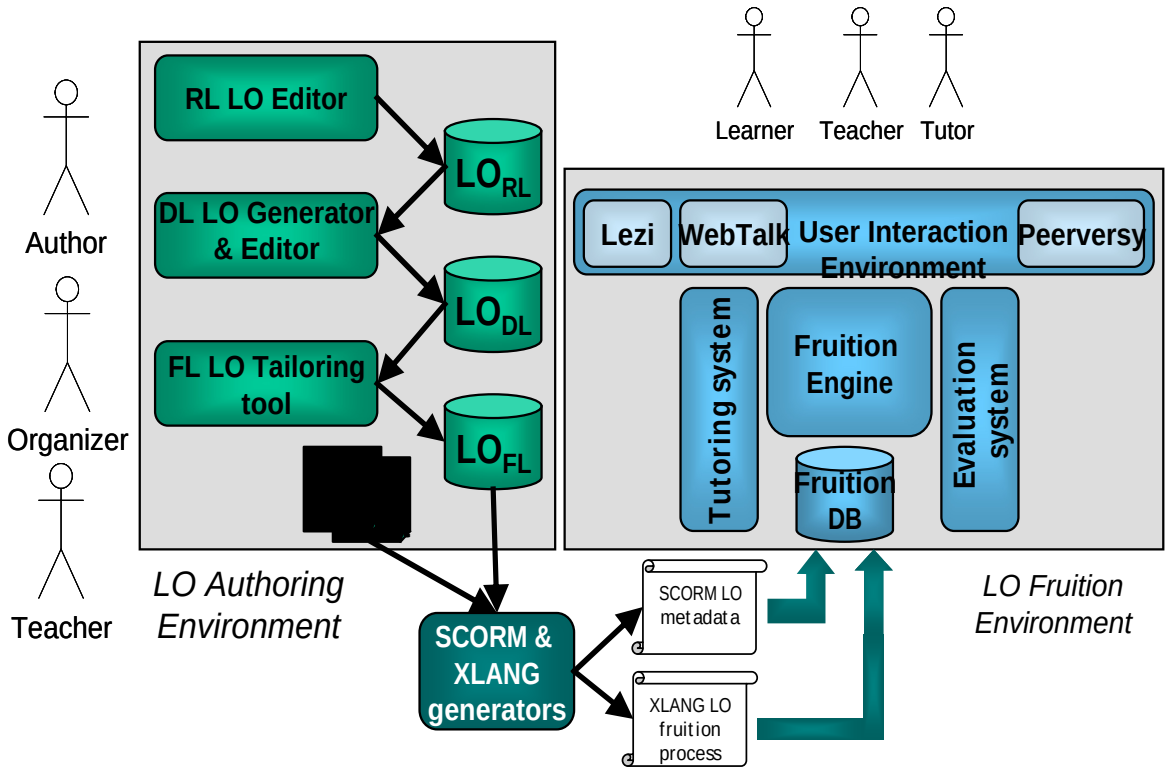
- Supportare il design, aggregazione e riuso di Learning Objects (LOs).
- Supportare la fruizione di LOs.
- Supportare l'analisi delle prestazioni e del comportamento dello studente.
- Analizzare il consumo di energia in ogni possibile situazione.
- Sperimentare l'utilizzo della piattaforma didattica nei corsi dell'ateneo.

La riusabilità dei contenuti didattici è un aspetto fondamentale per migliorare il rapporto costo/efficacia dei sistemi didattici informatici. Per questo motivo il materiale didattico nel progetto Virtual Campus prende il formato di "learning object"(LOs) avente arbitraria complessità, tipicamente organizzato in gerarchie di più piccola granularità e descritto da un modello di metadati. Una ricca descrizione semantica dei LOs è utile per capire le loro caratteristiche, permettere la loro pubblicazione in ambiente Web e supportare operazioni di ricerca avanzate. Il LO è la più piccola unità che può essere distribuita in Virtual Campus. E' la più piccola unità che può assegnata ad uno studente e per il quale lo studente può essere valutato. Fisicamente un LO è composto da una parte di metadati e una parte di contenuti. Nel modello di Virtual Campus ogni entità può essere classificata in termini di LOs. Oggetti come "lezione", "esame", "corso" vengono considerati sempre come LOs. LOs semplici possono essere aggregati in entità più complesse attraverso l'uso di "relazioni".

Un sistema in grado di gestire LOs dovrebbe poterli rappresentare mediante tre differenti livelli di astrazione: il livello del riutilizzo, dove il learning object è definito in modo da poter essere utilizzato in contesti diversi; il livello didattico, dove i LOs vengono visti in uno specifico conteso didattico; e il livello di fruizione dove viene a prendere forma il la lezione descrivendone i punti chiave, gli attori partecipanti i loro ruoli e così via.

L'architettura di VC è quindi composta da: un ambiente di creazione di contenuti didattici e un ambiente di fruizione di tali contenuti. L'ambiente di creazione permette agli utenti di creare e gestire LOs ai tre livelli precedentemente descritti. L'ambiente di fruizione mette a disposizione tutti i meccanismi per permettere la fruizione di LOs con scopi didattici. Il cuore dell'ambiente di fruizione è un motore che guida il processo di fruizione definito come parte del LO stesso. Il sistema di valutazione permette di raccogliere e analizzare i dati provenienti dall'interazione con gli studenti e con i docenti. Questi dati possono essere utilizzati come feedback anche per migliorare la qualità dei LOs.

La figura seguente propone visione schematica dell'architettura della piattaforma Virtual Campus. Si può notare come lezi.NET sia presente fra le applicazioni dedicate all'interazione con l'utente.



Per una visione maggiormente dettagliata riguardante l'architettura e il modelli concettuali della piattaforma Virtual Campus di rimanda a http://www.elet.polimi.it/res/vcampus/.

3 Standard

3.1 IMS

IMS[7] nasce come progetto del National Learning Infrastructure Initiative con lo scopo di fornire delle linee guida e delle specifiche che cerchino di definire quali siano i principali componenti ed entità necessarie per l'organizzazione ed implementazione di un sistema per la gestione e distribuzione di contenuti per le scuole superiore e le università.
Tuttavia le specifiche pubblicate sembravano essere valide per un campo di applicazione decisamente più vasto che quello della *higher education* riuscendo a soddisfare i requisiti imposti anche da altri tipi di organizzazione che necessitavano di un sistema organizzato ed omogeneo per la distribuzione di conoscenza come enti governativi ed organizzazioni private.
Il nome originale dell'iniziativa, da cui la sigla IMS, era Instructional Management System project. Tale definizione potrebbe far pensare ad un progetto vicino all'aspetto implementativo di un sistema per la gestione di corsi come un learning server, invece le specifiche e le linee guida prodotte riguardano standards che possono essere utilizzati per la progettazione di sistema di learning, per l'organizzazione dei contenuti e per facilitare l'interoperabilità fra sistemi nell'ambito di questo dominio.
Questo significa che diversi tipi di implementazione di sistemi di learning sia on-line (come i sistemi Web-based) che off-line (come la distribuzione di contenuti tramite media quali CD-ROM), che distribuiscono i contenuti in modo sincrono oppure asincrono, possono comunque trarre beneficio dalle specifiche e dalle linee guida proposte da IMS. Inoltre anche il campo di applicazione di sistemi di questo tipo spazia da realtà scolastiche, per le quali la realizzazione delle specifiche era stata originariamente pensata, fino a realtà aziendali e governative. Le specifiche IMS quindi permettono, grazie alla loro generalità e livello d'astrazione, di progettare sistemi altamente flessibili in grado di interoperare fra loro.
La generalità e livello d'astrazione delle specifiche IMS non compromette tuttavia l'aspetto operativo. Si possono infatti trovare, nei documenti prodotti da IMS, dei documenti formali che forniscono "interfacce" che possono fungere da confine fra i vari moduli che costituiscono una particolare implementazione di un sistema. Altri documenti, come le "*Best Practices*" e le *"Implementation Guide*", possono essere utilizzate dallo sviluppatore in modo che quest'ultimo possa concentrarsi sull'aspetto implementativo e tecnico senza dover reinventare ciò che il team di IMS ha già documentato e/o formalizzato.
La realizzazione delle specifiche IMS prevede una prima fase durante la quale si cerca di capire quali siano le esigenze fondamentali (soprattutto quelle di interoperabilità) per la realizzazione di una documentazione che possa utilizzata in ambito internazionale sempre mantenendo un giusto rapporto fra generalità ed operatività. Una volta che la specifica viene completata e sottoposta con successo a test (viene cioè utilizzata in una o più implementazioni reali), viene approvata formalmente dall' IMS Technical Board e rilasciata gratuitamente al pubblico.
Lo scopo finale è quindi quello di rendere omogeneo ed interoperabile il dominio dei sistemi di learning in modo da far fruire tutti gli attori dei benefici che una standardizzazione di questo tipo può portare.

[7] *Vedi IMS*

3.1.1 Specifiche prodotte

L'intero framework specificato dai documenti IMS risulta essere ampio e complesso. Al fine di ridurre la complessità e renderlo modulare è stato suddiviso in tre grandi sottoinsiemi: Content packaging, Data Model e Run-Time environment. Ognuno di questi sottoinsiemi può essere composto da un insieme di specifiche IMS. La Figura 3 fornisce una visuale completa del framework e delle relazioni che intercorrono fra le varie specifiche.

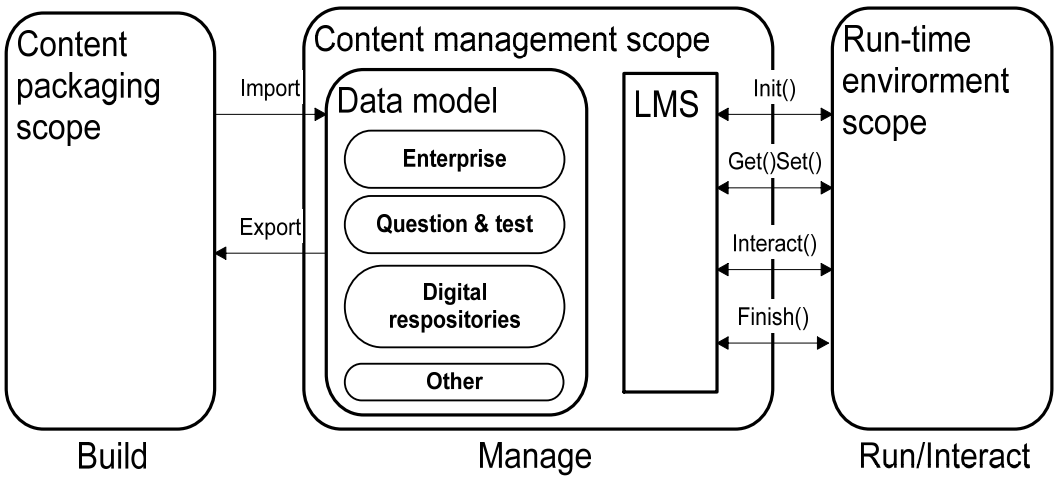


Figura 3: Relazioni fra vari scope di specifica.

Una singola specifica IMS è spesso composta da tre tipi di documento: un documento riguardante l'**information model** nel quale è descritta la specifica vera e propria, un documento che formalizza la specifica mediante l'utilizzo di XML, DTD e XML-Schema detto **XML Binding specification**, ed un documento che fornisce delle **best practices** ed **implementation guides** per poter facilitare la "messa in opera" della specifica stessa da parte degli sviluppatori.

- **Meta-data specification** : riguardano la definizione dei campi, le definizioni, il formato dei dati, le regole ed i controlli che sono necessari per la descrizione esaustiva e formale delle entità che possono essere presenti nei sistemi di learning. La specifica è basata su una versione modificata e approvata dall' IMS Technical Board di un documento IEEE[8] (Working Draft 6.1 Learning Object Meta-Data (LOM) Vedi IEEE.LOM). Il documento non è altro che una lista di metadati descritti ognuno da informazioni come nome, descrizione, molteplicità, tipo etc. Ad esempio l'elemento "Title" è descritto come "Nome del learning object", ha "single" come molteplicità e ha "LangStringType (1000 char)" come tipo. L'organizzazione dei metadati è di tipo gerarchico costituita da elementi che sono costituiti da sottoelementi.

- **Enterprise specification**: riguardano la definizione di alcune strutture dati, ad esempio l'entità User oppure Group. Tali definizioni dovrebbero poter garantire

[8] *Vedi IEEE*

una certa omogeneità ed interoperabilità fra le varie implementazioni, fornendo ad esempio la possibilità di scambio di informazioni relative ad utenti e gruppi di utenti fra sistemi diversi.

- **Content packaging specification**: descrivono le strutture dati con le quali i contenuti dovrebbero essere organizzati per garantire interoperabilità fra i LMS, i tools per la creazione dei contenuti ed i sistemi per la distribuzione di tali contenuti. Lo scopo ultimo di questa specifica è quello di fornire un set standardizzato di strutture per lo scambio di contenuti in modo che un corso prodotto da un sistema di creazione possa essere eseguito senza problemi d'incompatibilità da un altro sistema dedicato, ad esempio, all'esecuzione[9]. Vedi Figura 4 e Figura 5.

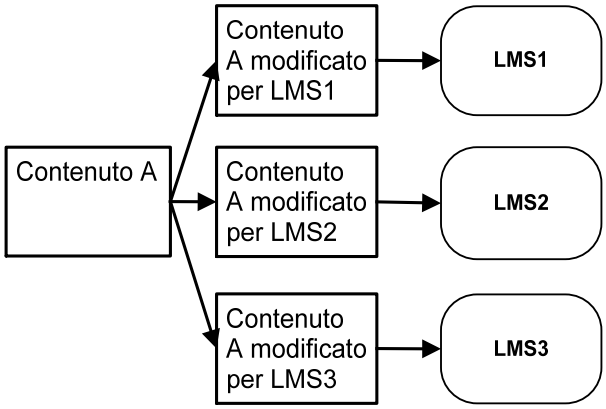


Figura 4: Senza interoperabilità.

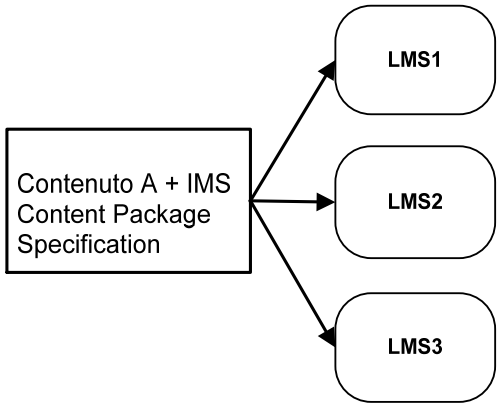


Figura 5: Interoperabilità garantita da specifiche IMS comuni ai tre LMS.

- **Question & Test Interoperability Specification:** riguarda la definizione di un linguaggio XML standard per la descrizione di domande e tests. Questa specifica può essere utile per la produzione di corsi in cui accanto alla parte di acquisizione dei contenuti è presente una parte che verifica l'effettivo grado di apprendimento degli utenti. Tools per la produzione di test possono pubblicare documenti XML che possono essere importati in altri tools di produzione di corsi e tali documenti XML possono essere correttamente interpretati dai sistemi

[9] *Tali sistemi sono spesso indicati come Run-Time environment .*

di distribuzione dei contenuti che, nel caso di tests, devono anche occuparsi di effettuare una registrazione delle prestazioni dell'utente sottoposto a verifica.

- **Digital Repositories Interoperability :** Definisce un insieme di API che ogni repository di contenuti dovrebbe implementare per poter fornire un accesso omogeneo a tutti gli altri sistemi che utilizzano e producono contenuti.

3.1.2 Content Packaging specification

Dell'insieme delle specifiche IMS, il progetto lezi.NET pur seguendo la maggior parte delle guide lines proposte, implementa solo le IMS Content Packaging Specification[10]. Per altri sottosistemi, come il Run-time environment, sono state seguite specifiche SCORM come sarà spiegato in seguito.
Nel contesto dell'e-Learning sono presenti entità quali testo, fotografie, immagini, animazioni, registrazioni audio e video etc.
Tali contenuti devono essere organizzati e pacchettizzati, primo per poter creare un corso composto da una raccolta di learning objects, secondo per poter avere in un unico pacchetto tutti i contenuti necessari per la fruizione del corso, infine per poter archiviare facilmente il corso in digital repositories permettendo un facile accesso ed un facile riutilizzo delle sue parti.
Inoltre, per poter garantire interoperabilità fra i vari LMS, è necessario definire un formato comune per la costruzione di questi corsi pacchettizzati. Per questi motivi IMS ha prodotto le specifiche di Content Packaging.
Tali specifiche permettono, infatti, a tutti i sistemi che le implementano di scambiarsi contenuti e di costruire, eseguire ed immagazzinare corsi. In pratica si tratta di un *package* che comprende principalmente due elementi: un documento XML chiamato manifest il cui nome del file deve essere **imsmanifest.xml** che descrive l'organizzazione dei contenuti e le risorse presenti nel package e dai files fisici che sono indicizzati dal manifest. L'insieme di questi files è incluso in un singolo file che rappresenta fisicamente il package. Tale file, detto **Package Interchange File**, deve contenere in radice il file XML imsmanifest.xml. Gli altri files, quelli indicati dal manifest, possono essere posizionati liberamente all'interno del package per quanto riguarda la struttura di directory, pur che siano raggiungibili utilizzando le informazioni, in pratica un percorso relativo, contenute nel manifest. Il package può essere un archivio di tipo *.cab, *.rar, *.jar ma è consigliato l'utilizzo del formato *.zip V2.04 o successiva. Riassumendo:

- **Package Interchange File:** un file .zip che contiene in radice il file imsmanifest.xml

- **Package:** una struttura logica di directory che contiene tutti i files che il manifest referenzia. Un package può essere una parte di un corso, un corso intero o anche una collezione di corsi. Ogni package dovrebbe contenere tutte le risorse necessarie per la sua esecuzione e contenere il minimo numero possibile di riferimenti esterni. Questo perché quando si ha una risorsa esterna ad un package, si rende il package dipendente dalla disponibilità di tale risorsa limitandone l'utilizzo. Non è necessario che il package sia sotto forma di

[10] *Per le specifiche complete vedere IMS.CP*

archivio zip. Un package potrebbe essere distribuito su di un CD-ROM senza essere compresso in un singolo file.

- **Top-level manifest:** un documento XML che descrive l'organizzazione dei contenuti e le risorse presenti nel package. Può, opzionalmente, contenere un (sub)manifest. Il manifest è composto dalle seguenti sottosezioni:

- Sezione **Meta-data**: contiene alcune informazioni relative all'intero manifest.
- Sezione **Organizations**: contiene le informazione relative all'organizzazione delle risorse del package. In pratica descrive il o i modi con cui le risorse verranno presentate all'utente del corso.
- Sezione **Resources**: contiene i riferimenti ai files fisici all'interno del package come files html, media, documenti etc, che costituiscono i contenuti veri e propri del corso.
- Sezione **(sub)Manifest**: eventuali sub manifest.

- Files fisici

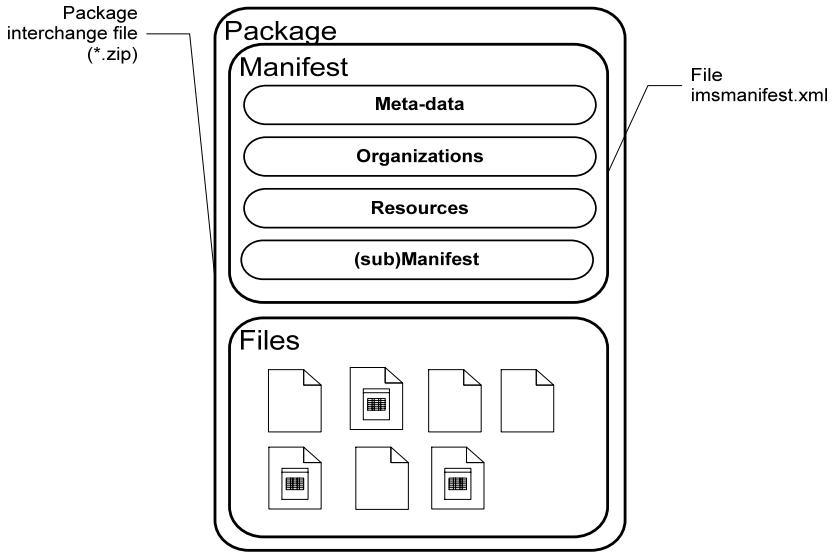


Figura 6:IMS Package

3.1.3 Struttura del manifest

Di seguito sarà presentato un elenco descrittivo e non completo della struttura di un manifest.
Per una descrizione formale si rimanda a `http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1p3.xsd` e `http://www.imsglobal.org/xsd/imsmd_v1p2 imsmd_v1p2p2.xsd`.
Se il documento XML manifest è *well-formed* e rispetta i precedenti XML-Schema definition può essere ritenuto formalmente corretto.

- **<manifest>** (1)[11] : è l'elemento radice del documento XML. Contiene tre sub elementi:

[11] *La notazione fra parentesi indica la cardinalità dell'elemento. (1) : una istanza obbligatoria, (0-1): zero o una istanza, (0:n): zero, una o n istanze, (1:n) una o n istanze.*

- <**metadata**> (0-1) : descrive il manifest stesso. Può includere elementi come titolo, descrizione, scopo dei contenuti etc.
- **<organizations>** (1): contiene zero, una o più descrizioni statiche su come è organizzata la presentazione delle risorse.
- **<organization>** (0:n): rappresenta un singola organizzazione dei contenuti. In questa versione delle specifiche l'elemento organization possiede l'attributo di default hierarchical poiché è possibile organizzare la presentazione delle risorse solo mediante una rappresentazione ad albero. Future versioni delle specifiche prevedranno la possibilità di presentare i contenuti anche mediante diversi tipi di forme come reti semantiche e grafi condizionali.
- <**item**> (1:n) : indica un nodo dell'albero di rappresentazione. Ogni elemento item può contenere altri nodi item.
- <**title**> (1) : titolo del nodo.
- **<resources>** (1) : contiene un elenco di elementi resource necessari per poter visualizzare i contenuti come indicato nell'elemento organizations.
- **<resource>** (0:n) : rappresenta una singola risorsa. Conterrà elementi con riferimenti ai files della risorsa.
- **<metadata**> (0-1) : metadati che descrivono la risorsa.
- <**file**> (0:n) : ogni elemento resource può contenere diversi elementi file che contengono informazioni relative ad uno dei file fisici della risorsa.
- <**meta-data>** (0-1) : meta-dati descrittivi del file.
- <**dependency**> (0:n) : indica la dipendenza, a livello di contenuti, di una risorsa da altre risorse.

Elementi come *organization*, *item* e *resource* possiedono l'attributo *identifier* il cui valore indica l'identificatore univoco dell'elemento. Tale univocità è relativa e valida solo all'interno del manifest stesso. L'elemento *item* possiede, oltre che l'attributo *identifier*, anche l'attributo *residentifier* il cui valore permette di poter puntare univocamente alla risorse che gli item rappresentano. Un *item* senza una risorsa corrispondente avrà una funzione esclusivamente organizzativa.

Esempio 1: elemento organization che definisce la struttura gerarchica con cui verranno presentate le risorse.

```
<organization identifier="TOC1">
  <title>Default organization</title>
  <item identifier="ITEM1" identifierref="RESOURCE1">
    <title>Lesson 1</title>
  </item>
            <item identifier="ITEM11" identifierref="RESOURCE11">
                  <title>Lesson 1.1</title>
            </item>
    <item identifier="ITEM2" identifierref="RESOURCE2">
      <title>Lesson 2</title>
    </item>
  <item identifier="ITEM3" identifierref="RESOURCE3">
    <title>Lesson 3</title>
  </item>
</organization>
```

Un LMS che leggerà l'elemento organization *dell'*

Esempio 1 lo interpreterà concettualmente nel seguente modo:

- Lesson 1
  - Lesson 1.1
- Lesson 2
- Lesson 3

Attivando una delle voci l'LMS avvierà la risorsa relativa visualizzandola nell'ambiente di esecuzione.

Esempio 2: elemento resources del manifest che contiene una lista di elementi resource. L'esempio è significativo dal punto di vista sintattico.

```
<resources>
  <resource identifier="RESOURCE1" type="webcontent" href="sco01.html">
    <metadata/>
    <file href="sco01.html" />
    <file href="scripts\APIWrapper.js" />
    <file href="scripts\Functions.js" />
    <dependency identifierref="RESOURCE2" />
    <dependency identifierref="RESOURCE3" />
  </resource>
  <resource identifier="RESOURCE11" type="webcontent" href="sco011.html">
    <metadata/>
    <file href="sco011.html" />
    <file href="scripts\APIWrapper.js" />
    <file href="scripts\Functions.js" />
    <dependency identifierref="RESOURCE1" />
  </resource>
  <resource identifier="RESOURCE2" type="webcontent" href="sco02.html">
    <metadata/>
    <file href="sco2.html" />
    <file href="scripts\APIWrapper.js" />
    <file href="scripts\Functions.js" />
  </resource>
  <resource identifier="RESOURCE3" type="webcontent"
href="pics\distress_sigs.jpg">
    <metadata/>
    <file href="pics\distress_sigs.jpg" />
  </resource>
</resources>
```

Esempio 3: documento manifest (non intero).

```
<?xml version="1.0"?>
<manifest identifier="MANIFEST1" xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
    xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_v1p2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1p3.xsd
http://www.imsglobal.org/xsd/imsmd_v1p2 imsmd_v1p2p2.xsd">
             <metadata>
                   <schema>IMS CONTENT</schema>
                   <schemaversion>1.1</schemaversion>
                   <imsmd:lom>
                   <imsmd:general>
                   <imsmd:title>
                   <imsmd:langstring xml:lang="en">Mastering Visual Basic 6 -
Sample Chapter</imsmd:langstring>
             </imsmd:title>
             </imsmd:general>
```

```
            </imsmd:lom>
            </metadata>
<organizations>
            <organization identifier="TOC1">
                  <item identifier="LRNID44368448" identifierref="FILE14">
                        <title>Overview</title>
                        <item identifier="LRNID44368544"
identifierref="FILE16">
                              <title>Course Introduction</title>
                        </item>
                        <item identifier="LRNID44368640"
identifierref="FILE17">
                              <title>Copyright Information</title>
                        </item>
                  </item>
                  <item identifier="LRNID44368736" identifierref="FILE19">
                  <title>Chapter 5 Using ActiveX Data Objects</title>
                        <item identifier="LRNID44368832"
identifierref="FILE21">
                              <title>Overview of ADO</title>
                        <item identifier="LRNID44368928"
identifierref="FILE23">
                              <title>Introduction to ADO</title>
                        </item>
. . .
            </organization>
</organizations>
<resources>
            <resource identifier="FILE14" type="webcontent" href="course.htm">
                  <file href="course.htm"/>
                  <file href="Courseimages/intjump.gif"/>
                  <file href="Courseimages/MVB6Logo.gif"/>
            </resource>
            <resource identifier="FILE16" type="webcontent" href="course_1.htm">
                  <file href="course_1.htm"/>
                  <file href="Courseimages/asf_100.gif"/>
                  <file href="Courseimages/asf_56.gif"/>
                  <file href="Courseimages/avi_download.gif"/>
                  <file href="Courseimages/exppov.gif"/>
                  <file href="Media/mvb6_00p005-100.asx"/>
                  <file href="Media/mvb6_00p005-56.asx"/>
                  <file href="Media/mvb6_00p005.avi"/>
            </resource>
            <resource identifier="FILE17" type="webcontent"
href="copyright.htm">
                  <file href="copyright.htm"/>
            </resource>
            <resource identifier="FILE19" type="webcontent"
href="MVB9800913.htm">
                  <file href="MVB9800913.htm"/>
            </resource>
            <resource identifier="FILE21" type="webcontent"
href="MVB9800914.htm">
                  <file href="MVB9800914.htm"/>
            </resource>
            <resource identifier="FILE23" type="webcontent"
href="MVB9800915.htm">
                  <file href="MVB9800915.htm"/>
            </resource>
. . .
</resources>
</manifest>
```

3.2 ADL

L'iniziativa Advanced Distributed Learning[12] (ADL), voluta dal Dipartimento della Difesa (DoD) americano nel 1997, è un patto di collaborazione fra industria, istituzione governative e scolastiche per definire un nuovo ambiente di sviluppo comune per l'e-learning al fine di garantire interoperabilità fra i vari sottosistemi quali LMS, tools di sviluppo, di produzione, di editing e riusabilità, accessibilità e durabilità dei contenuti prodotti.
Nel Gennaio 2000 ADL pubblica una prima versione di SCORM (Sharable Content Object Reference Model) un insieme di specifiche e standards ottenute come unione e revisione di vari gruppi di standards dominanti (Vedi Figura 7) in risposta agli obiettivi che il team ADL si era proposto. Migliorata in base alle esperienze implementative della versione 1.0, un anno più tardi viene rilasciata la versione 1.1. Infine, data la stabilità raggiunta nella versione 1.1, viene prodotta la 1.2 che integra delle specifiche di content packaging concepite come estensione delle corrispondenti specifiche IMS[13]. Vengono inoltre estese anche le IMS Meta-data specification per una migliore descrizione dei contenuti.

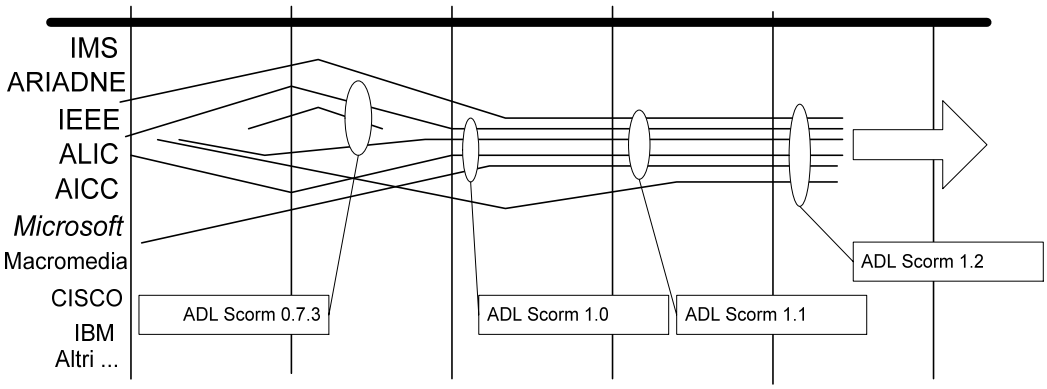


Figura 7: Convergenza delle specifiche e degli standards.

3.2.1 SCORM 1.2

SCORM 1.2[14] è un insieme di specifiche costituito da tre libri. La struttura della documentazione è la seguente:

- **Overview**: contiene una introduzione riguardante l'iniziativa ADL e le specifiche SCORM e la lista di specifiche tecniche e linee guida che costituiscono le sezioni seguenti.

- **Content Aggregation Model**: contiene specifiche per la definizione di un formato e di strutture dati che permettano l'aggregazione di risorse (ad esempio documenti, immagini, video etc) in contenuti di e-learning strutturati (ad

[12] *Vedi ADL.*
[13] *Le IMS Content packaging sono a loro volta derivate dalle specifiche IEEE Learning Technology Standards Committee (LTSC) (Vedi IEEE.LOM) prodotte mediante sforzi condivisi di IMS stessa e Alliance of Remote Instructional Authoring and Distribution Networks for Europe (Vedi ARIADNE).*
[14] *Per approfondimenti sulle specifiche SCORM vedi: ADL.SCORM.OW, ADL.SCORM.CAM, ADL.SCORM.RTE.*

esempio corsi, lezioni etc). Il documento è composto dalle tre seguenti sottosezioni:

- Meta-data dictionary (da IEEE)
- Content packaging (da IMS + estensioni)
- Content aggregation components

- **Run-time environment**: contiene specifiche che definiscono un ambiente (generalmente Web-based) che permetta l'esecuzione, la comunicazione con un LMS ed il tracking degli eventi e dati quali tempi di esecuzione, risultati di tests etc. Queste specifiche sono derivate tenendo conto delle funzionalità del run-time environment specificato nelle CMI001 guidelines di AICC (Vedi AICC-CMI). Il documento è composto dalle due seguenti sottosezioni:

  - Data Model (da AICC)
  - Run-time API (da AICC)

3.2.2 Content aggregation model

Le specifiche descritte da questo documento permettono di creare dei contenuti ottenuti mediante l'aggregazione di risorse. E' stata necessaria quindi la definizione di una nomenclatura per poter identificare e aggregare certi tipi di risorse in base alle loro caratteristiche. Nel CAM[15] sono stati definiti tre tipi di entità principali.

- **Assets**: sono l'elemento elementare nel dominio dell'e-learning. Possono essere visti come la rappresentazione elettronica di testi, immagini, suoni, pagine Web etc.
- **Sharable content object (SCO):** è una collezione di uno o più assets che generalmente include uno specifico asset (ad esempio una pagina html con codice Jscript) che permette l'utilizzo del Run-Time environment per la comunicazione con l'LMS. Lo SCO rappresenta il più basso livello di granularità che una risorsa di learning, interagente tramite le API con il LMS, possa avere. Per garantire la riusabilità, uno SCO dovrebbe essere indipendente dal contesto in cui viene creato, in modo che possa essere riutilizzabile in altri contesti insieme ad altri SCO creati in contesti ancora diversi. L'interindipendenza fra SCO impone un ulteriore vincolo: uno SCO non deve mai fare riferimento ad un altro SCO tramite, ad esempio, un hyperlink. Questo perché se uno SCO potesse lanciare un altro SCO verrebbe scavalcato il meccanismo di *launch* dell'LMS compromettendo il corretto funzionamento del Run-Time environment. SCORM non impone limiti alle dimensioni degli SCO, tuttavia consiglia di suddividere un gran contenuto in unità logicamente indipendenti e di creare diversi SCO per ognuna di queste unità e, infine, di aggregare gli SCO prodotti per riottenere il contenuto iniziale.

[15] *Content aggregation model.*

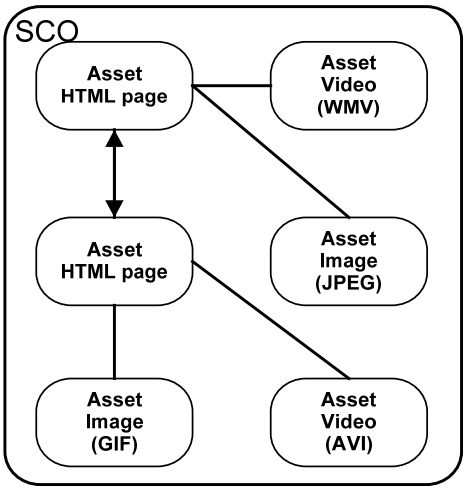


Figura 8: Esempio di SCO. All'interno dello SCO sono presenti diversi assets. La navigazione fra i diversi assets è possibile ed è responsabilità dello SCO fornire gli hyperlink necessari.

- **Content aggregations:** sono una collezione di assets e di SCO aggregati per poter costruire una risorsa di learning ad alto livello come ad esempio un corso. Un pacchetto, ad esempio un archivio zip, contenente un manifest e diversi assets è una possibile rappresentazione fisica di una content aggregation.

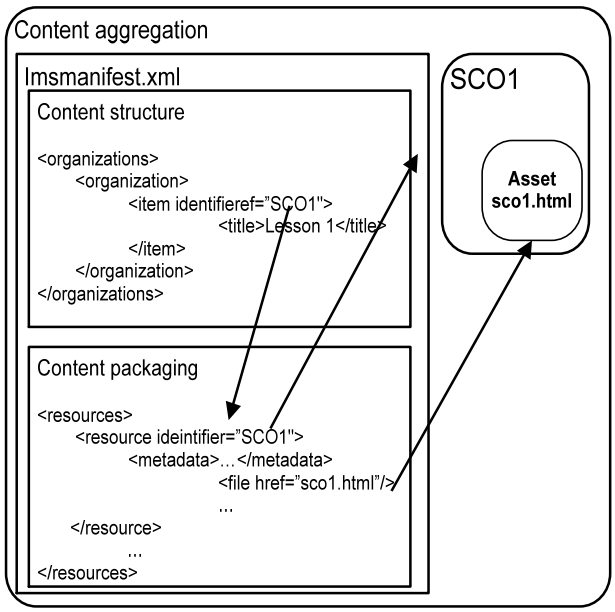


Figura 9: Content aggregation contenente uno SCO con un asset. Il manifest descrive la struttura interna della content aggregation. La content structure descrive il modo in cui i contenuti verranno presentati, le Content packaging forniscono tutti i riferimenti agli SCO a gli Assets.

Per poter formalizzare tali operazioni di aggregazione, partendo dalla risorsa più elementare fino ad arrivare ad un content aggregation, sono necessarie:

- Le Content Packaging, estensione delle specifiche IMS Content Packaging viste nel Capitolo 3.1.2, permettono di creare le content aggregations definendo inoltre l'organizzazione dei contenuti. La rappresentazione fisica consiste nel file imsmanifest.xml che contiene le informazioni relative alla organizzazione delle risorse e ai riferimenti ai files degli assets.

- Le Meta-Data dictionary che definiscono dei meta-dati per ognuna delle tre precedenti entità in modo da facilitare le operazioni di archiviazione, ricerca e riutilizzo delle risorse ad ogni livello di granularità. I meta-dati vengono rappresentati mediante elementi presenti nel manifest e documenti xml stand-alone associati a ciascuno SCO. Un riferimento a tali files è comunque presente nel manifest.

3.2.3 Run-Time environment

Al fine di poter garantire il riutilizzo delle risorse e l'interoperabilità fra diversi LMS è necessario definire un unico ambiente d'esecuzione che fornisca una interfaccia comune per poter permettere l'esecuzione delle risorse e la comunicazione di quest'ultime con il LMS. Il Run-Time environment fornisce tre tipi di entità per soddisfare tali requisiti: un meccanismo di lancio, *launch*, una interfaccia di comunicazione, *API*, ed un modello di dati, *data model*. Nella Figura 9 è possibile osservare le relazioni che intercorrono fra le suddette entità.

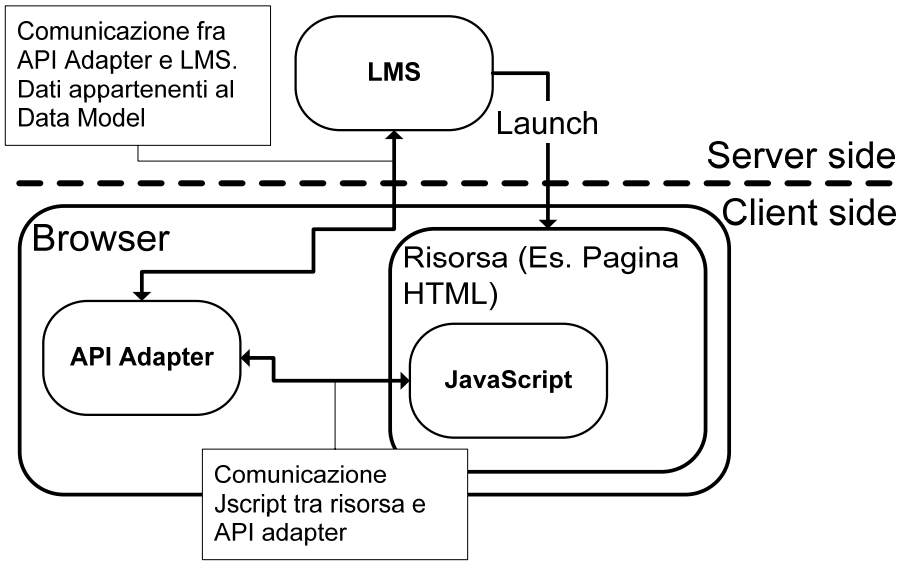


Figura 10: Interazione fra LMS e risorsa.

Il meccanismo *launch* permette il lancio, inteso come messa in esecuzione, di una risorsa (SCO) in un ambiente, generalmente Web-based. La comunicazione fra la risorsa in esecuzione ed il LMS e viceversa avviene mediante una API che fornisce un numero limitato di metodi per l'interazione con il *Data Model* e per la trasmissione di informazioni riguardanti lo stato della risorsa (es. Inizializzata, terminata o in errore) che vengono tenute in considerazione dal LMS per gestire il corso in esecuzione.

3.2.4 Interfaccia di comunicazione (API)

L'utilizzo di un'interfaccia comune oltre a garantire interoperabilità e riusabilità delle risorse premette si semplificare il processo d'implementazione di un sistema di learning completo. Infatti, lo sviluppatore dei contenuti non dovrà preoccuparsi dei dettagli implementativi del LMS e, viceversa, lo sviluppatore di un LMS dovrà limitarsi a fornire, nell'ambiente di esecuzione, le API.

L'API Adapter è una porzione di codice che implementa le API è che viene fornita alla risorsa dal Run-Time generalmente mediante un oggetto JScript, con identificatore "API", che viene cercato nella struttura gerarchica dei frames del browser dalla risorsa stessa, dopo che è stata lanciata dal LMS.
Tale modulo può essere considerato come parte client-side del LMS eseguito nel browser.
Una volta che uno SCO è stato lanciato, tramite le API, potrà chiamare un insieme di otto metodi che avranno una duplice funzione:

- Gestione dello stato dello SCO e delle situazioni d'errore.
- Trasferimento di dati da e verso l'LMS.

Gli otto metodi sono i seguenti:

- **LMSInitialize:** lo SCO esegue questa chiamata per indicare che è pronto per l'esecuzione e permette al LMS di inizializzare la propria struttura dati (proprietaria) ed il Data Model per poter supportare l'esecuzione. E' obbligatorio che una SCO esegua questa chiamata prima di qualsiasi altra.
- **LMSFinish:** lo SCO esegue questa chiamata quando ritiene che il canale di comunicazione con il LMS non sia più necessario e che la propria esecuzione possa essere terminata
- **LMSGetValue:** questo metodo permette allo SCO di ottenere informazioni dal Data Model gestito dal LMS.
- **LMSSetValue:** questo metodo permette allo SCO di scrivere nel Data Model.
- **LMSCommit:** lo SCO chiama questo metodo quando desidera che le precedenti operazioni di scrittura (mediante metodo LMSSetValue) debbano essere considerate valide e salvate in modo da garantire persistenza.
- **LMSGetErrorString:** permette di ottenere una descrizione testuale di un errore espresso in formato numerico passato come parametro al metodo.
- **LMSGetLastError:** ritorna un numero indicante un'eventuale condizione d'errore rilevata dal LMS durante l'esecuzione dello SCO e quindi delle varie chiamate all' API che quest'ultimo può aver fatto.
- **LMSGetDiagnostics:** necessario per poter supportare una descrizione degli errori dettagliata, specifica di un particolare LMS (mentre i tipi d'errori ritornati da LMSGetErrorString sono appartenenti allo standard).

I metodi LMSGetValue e LMSSetValue permettono, rispettivamente, di leggere e scrivere valori nei campi del Data Model. Per poter indicare quale campo del Data Model andare a leggere o scrivere, è passata, come parametro del metodo, una stringa espressa in *dot-notation*. Ad esempio, lo SCO per poter ottenere il nome dello studente che sta eseguendo il contenuto ed eventualmente visualizzarlo nella pagina esegue una chiamata di questo tipo:

```
var name = LMSGetValue("cmi.core.student_name ");
```

La stringa `cmi.core.student_name`[16] indica un campo del Data Model inizializzato probabilmente dal LMS con il nome dello studente che ha chiesto il lancio dello SCO. L'utilizzo di questo *modus operandi* permette sia di ridurre a due il numero di metodi necessari per poter avere accesso ad un ampio Data Model, sia di rendere indipendenti le API dal Data Model stesso. Infatti, un cambiamento del modello, probabilmente un'estensione, non rende necessaria alcuna modifica all'API.

[16] *Una spiegazione dettagliata del Data Model verrà fornita nel sotto capitolo 3.2.5*

Una soluzione di questo tipo introduce, tuttavia, una maggiore complessità implementativa al livello di LMS in quanto sarà necessaria la realizzazione di un parser che leggendo la stringa in *dot-notation* permetta di indirizzare correttamente i campi del Data Model.

Lo stato viene invece gestito mediante i metodi LMSInitialize, LMSCommit e LMSFinish. La possibilità, da parte dello SCO, di invocare le chiamate LMS è vincolata dallo stato in cui si trova lo SCO stesso. In Figura 11 e possibile vedere gli stati e le corrispondenti chiamate LMS possibili.

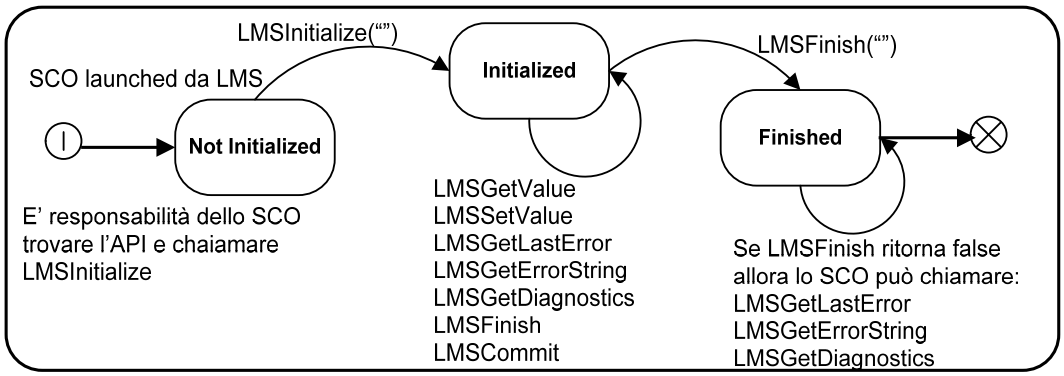


Figura 11: transizioni di stato dello SCO.

- **NotInitialized:** è lo stato in cui si trova lo SCO fra il momento del lancio da parte del LMS ed il momento in cui lo SCO stesso chiama LMSInitialize. Durante questo stato lo SCO cerca nella struttura gerarchica dei frames del browser l'oggetto JScript fornito dal frame radice provvisto, a sua volta, dall'LMS.
- **Initialized:** è lo stato in cui si trova lo SCO dopo la chiamata LMSIntitalize e LMSFinish. In questo stato è possibile chiamare un qualsiasi metodo dell'API tranne LMSInitialize.
- **Finished:** è lo stato in cui lo SCO si trova dopo aver chiamato LMSFinish. Se il valore ritornato dall'API è "false" allora lo SCO può chiamare i metodi per gestione dell'errore. Vedi Figura 11.

3.2.5 Data model

Lo scopo di avere un modello di dati comune è quello di permettere agli SCO di interagire con diversi LMS. Se, per esempio, è necessario registrare le prestazioni di uno studente è indispensabile un sistema che permetta di informare l'LMS dei risultati ottenuti durante la fruizione dello SCO.
Gli SCO possono avere un'implementazione proprietaria delle strutture-dati necessarie per poter gestire i loro funzionamento, è necessario, tuttavia, una modello comune per poter comunicare con l'LMS scrivendo e leggendo dati utili per l'esecuzione. L'LMS potrà, poi, utilizzerà tale struttura dati comune oppure potrà tradurla in qualche altra rappresentazione proprietaria comoda per la particolare implementazione.
Il data model di SCORM è derivato direttamente AICC CMI Data Model descritto in AICC CMI Guidelines for Interoperability (Vedi AICC-CMI). Si tratta di una struttura ad albero la cui radice si chiama "cmi" per indicare che le foglie appartenenti a tale albero sono parte dell'AICC CMI Data Model in modo che future estensione del modello siano facilmente

realizzabili. Ad esempio, se ADL volesse inserire nel modello ulteriori metadati potrebbe utilizzare una diverso prefisso: *adl.elementName* invece di *cmi.elementName* . Parte delle struttura del modello può essere vista in Figura 12.

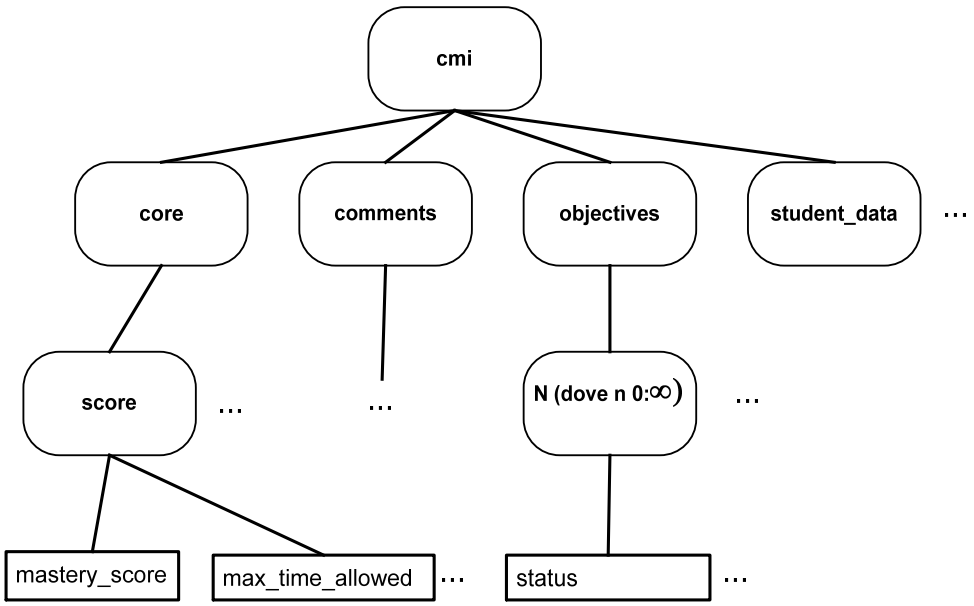


Figura 12: spaccato della struttura-dati CMI usata dal Run-Time SCORM per lo scambio di dati fra SCO in esecuzione e LMS. Un'istanza della struttura è identificata dalla coppia utente e SCO in esecuzione per quell'utente.

Ogni foglia della struttura ha un tipo di dato. Ad esempio, la foglia identificata dal percorso *cmi.core.lesson_status*, ha come tipo CMIVocabularyStatus che è un'enumerazione di stringhe: *passed, completed, failed, incomplete browsed, not attempted*. Per ogni foglia è inoltre definita anche l'accessibilità da parte dello SCO (l'LMS ha sempre accesso a tutti i campi). La foglia *cmi.core.lesson_status* ha *read/write*. Questo significa che lo SCO può leggere e scrivere il proprio stato durante la propria esecuzione nel Run-Time rispettivamente eseguendo le chiamate all'API:

```
var status = API.LMSGetValue("cmi.core.lesson_status"); // ritorna lo stato
API.LMSSetValue("cmi.core.lesson_status","completed);   // scrive lo stato
```

Se, ad esempio, uno SCO implementa un test a risposta multipla da sottoporre all'utente, una volta eseguito e determinato il risultato (generalmente tramite logica di controllo in JScript) questo può scrivere nel modello il suo esito per l'utente corrente mediante:

```
API.LMSSetValue("cmi.core.lesson_status","passed");
API.LMSCommit("");  // dice all'LMS di rendere persistente il lavoro fin qui
// eseguito
```

L'LMS che gestisce l'esecuzione dello SCO avrà precedentemente inizializzato la struttura CMI con i dati dello SCO e dell'utente che ne identificano l'istanza, leggendo il campo *cmi.core.lesson_status* dopo la chiamata LMSFinish("") saprà che l'utente ha passato il test.

Esistono naturalmente nella struttura CMI anche i metadati necessari per sapere le singole risposte dell'utente.

Non tutti i campi del modello presenti nelle specifiche richiedono un'implementazione da parte dell'LMS. Esistono, tuttavia, campi che devono obbligatoriamente essere implementati in modo da poter affermare che l'LMS sia compatibile, ad esempio, con SCORM 1.2. Per permettere agli SCO di "capire" quali campi siano effettivamente presenti nel modello, sono presenti dei campi speciali accedibili in sola lettura che contengono la lista dei metadati disponibili nel nodo. Ad esempio il nodo *cmi.core* possiede una foglia *cmi.core._children* che contiene la lista degli altri nodi/foglio alla stessa profondità*: student_id, student_name, lesson_location, credit, lesson_status, entry, score, total_time, lesson_mode, exit, session_time*. Una foglia *_children* è quindi presente in ogni nodo del modello.
Altri campi speciali sono *_count* utilizzato per poter sapere il numero di elementi presenti in un nodo in a cui è possibile aggiungere run-time altri nodi e *_version* utilizzato per determinare la versione del data model supportato dall'LMS.
Per una descrizione dettagliata del modello si rimanda a ADL.SCORM.RTE.DM.

Un esempio di specifica di un campo del data model è dato in Figura 13.

cmi.student_preference.language	

Supported API calls:
LMSGetValue()
LMSSetValue()
 	
LMS Mandatory:
No
 	
DataType:
CMIString255
 	
SCO Accessibility:
Read
Write
	Definition: For SCOs with multi-lingual capability, this element should be used to identify in what language the information should be delivered.
	
	Usage: Used by the SCO to both set and obtain from the LMS language preferences for the student.
	
	Format: Alphabetic string, may include white space.
	
	LMS Behavior:
	Initialization: If supported the LMS should initialize this value to an empty string (“”). It is the responsibility of the SCO to set this value. If an LMSGetValue() is requested before the SCO has set this value, then the LMS should return an empty string (“”)
	LMSGetValue(): Returns the current value stored by the LMS.
	Error Code:
	401 - Not implemented error. If element is not supported an empty string is returned and an error code is set to indicate that the element is not supported.
	LMSSetValue(): Sets the LMS data item to the value passed in as a parameter.
	Error Code:
	405 – Incorrect Data Type: If the element is supported and a request attempts to invoke an LMSSetValue() with a value that is not of the correct data type.
	401 - Not implemented error. If this element is not supported an error code is set to 401by the LMS to indicate that the element is not supported.
	Example Return/Set Values:
	"English"
	"French"
	
	SCO Usage Example:
	`LMSSetValue(“cmi.student_preference.language”, “English”);`
	`var languageValue = LMSGetValue(“cmi.student_preference.language”);`

Figura 13: Specifica del campo cmi.stundet_preference.language. Tale campo è di solito inizializzato dall'LMS che prende la preferenza dell'utente dal sottosistema di gestione dei profili degli utenti. CMIString255 è un tipo di dato, precisamente un stringa la cui lunghezza è al massimo di 255 caratteri.

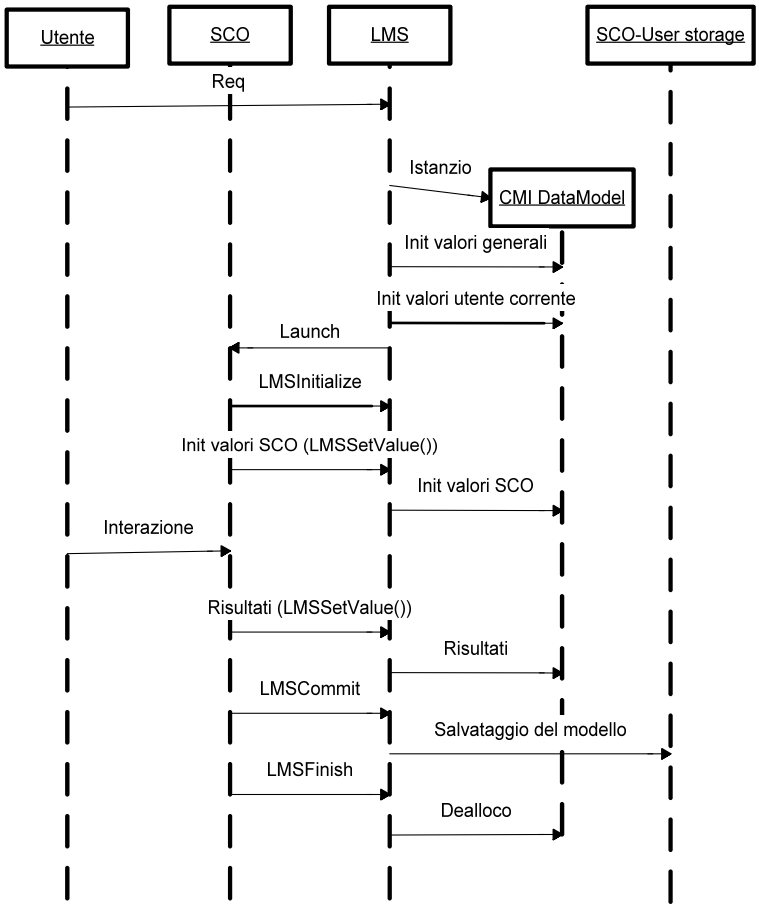


Figura 14: digramma rappresentante una possibile sequenza di interazioni fra utente, SCO, LMS e struttura CMI.

La Figura 14 riprende l'esempio precedente e descrive la sequenza di interazioni fra utente, SCO, LMS e struttura-dati CMI. L'utente esegue, ad esempio tramite interfaccia Web-based, la richiesta di lancio del prossimo SCO e di uno SCO specifico (se possibile). Il sistema, se accetta tale richiesta, inizia una serie di operazioni compresa la creazione del data-model CMI del quale creerà una nuova istanza per l'utente (che ha fatto la richiesta) e per lo SCO (che sta per essere lanciato). Durante questa fase l'LMS scrive dei dati nei campi di un modello "pulito" come: nome dell'utente, data e ora de lancio dello SCO, modalità di lancio dello SCO etc. In seguito, lancia lo SCO nell'ambiente d'esecuzione. Quest'ultimo inizierà la comunicazione mediante LMSInitialize(). Da questo momento in poi potrà interagire con l'LMS utilizzano i metodi dell'API LMSSetValue e LMSGetValue. Quando l'interazione fra utente e SCO è terminata quest'ultimo chiama LMSCommit e LMSFinish. Il sistema avrà quindi a disposizione un modello che descrive la storia e lo stato dell'esperienza di learning fatta dall'utente che verrà salvato per poter permettere ad esempio una ripresa dell'esecuzione dopo un'interruzione oppure per poter effettuare un valutazione delle prestazioni dell'utente.
Un possibile scenario che descrive la comunicazione fra SCO e LMS è rappresentato in Figura 15. Da rilevare che tale scenario e solo un esempio e non un'implementazione richiesta dalle specifiche.

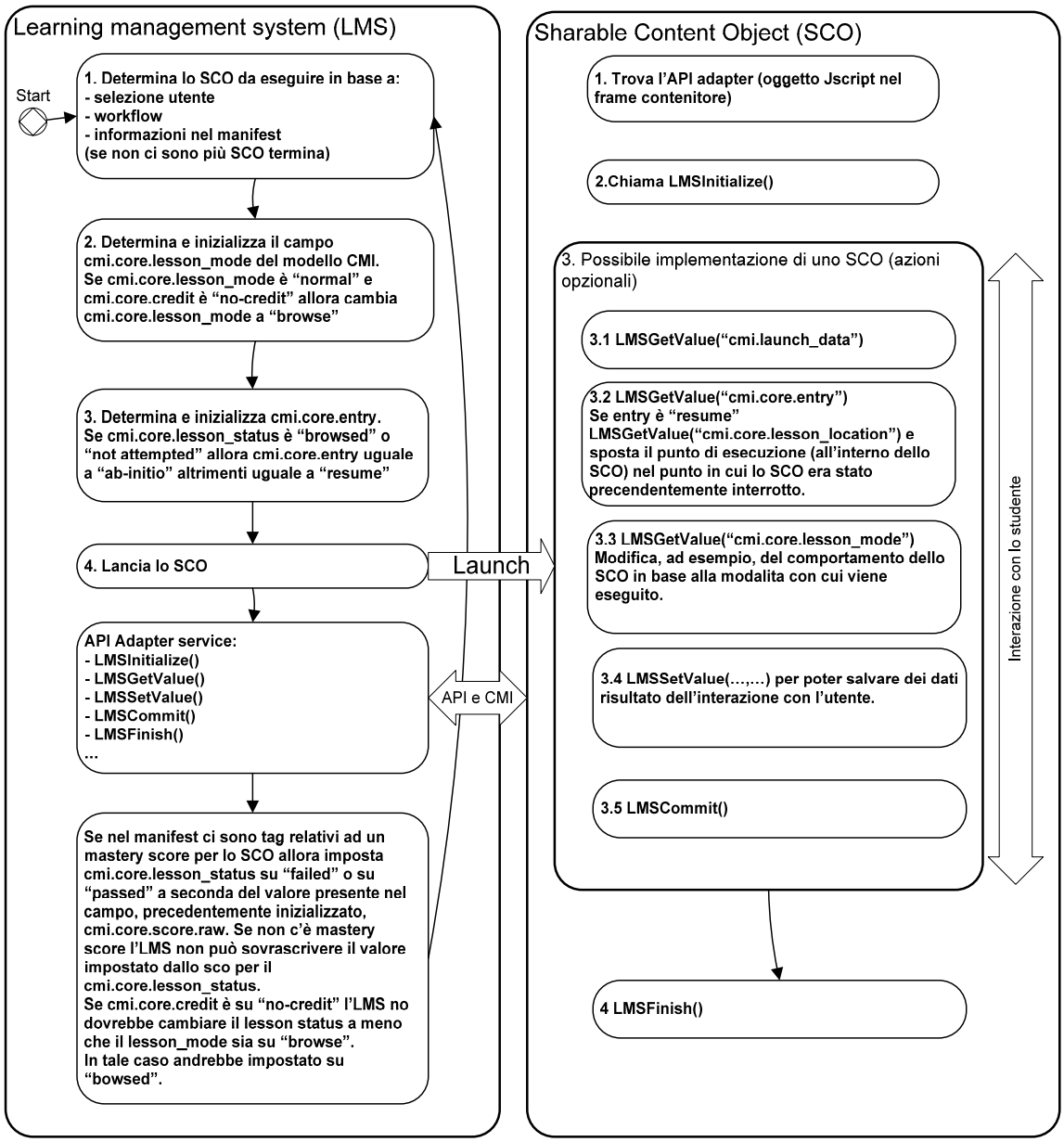


Figura 15: Possibile scenario implementativo di azione/interazione fra SCO e LMS.

4 Tecnologie

4.1 SOAP - WSDL– UDDI

SOAP[17] è un protocollo di comunicazione basato sullo scambio di documenti XML. E' stato originariamente sviluppato da Microsoft, IBM, Developer Mentor e Userland Software e, in seguito, sottoposto all'ITEF[18] che ne ha fatto uno standard ufficiale. La principale motivazione che ha portato allo sviluppo di questo protocollo, è stata la grande necessità di protocolli di comunicazione inter-applicazione. Protocolli come DCOM, RMI di Java, IIOP hanno permesso di pubblicare servizi in ambiente distribuito, tuttavia richiedono che i client sia complessi e che siano in grado di riprodurre tramite un proxy i servizi offerti dal server. Inoltre i loro limiti, riguardanti soprattutto sicurezza e facilità di deployment, divengono evidenti quando lo strato di comunicazione invece di essere una LAN o una WAN è Internet. SOAP è quindi una soluzione semplice per permettere l'interazione fra differenti applicazioni, scritte in differenti linguaggi e funzionanti su diverse piattaforme in quanto usa http per il trasporto e l'xml per codificare il carico pagante dei messaggi spediti e ricevuti.
I messaggi sono impachettati in quello che viene chiamato SOAP envelop (Vedi Esempio 4) e spediti/ricevuti da un server utilizzando il meccanismo Request/Response. Al contrario di DCOM e RMI, SOAP non richiede un connessione persistente fra il server ed il client.

Esempio 4. Scheletro di un messaggio SOAP.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
  ...
  ...
</soap:Header>
<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

Esempio 5. Messaggio SOAP relativo alla Request e la Response per il metodo GetStockPrice avente StockNamecome parametro e Price come valore di ritorno.

Request:

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
```

[17] *SOAP : Simple Object Access Protocol.*
[18] *ITEF: Internet Engineering Task Force.*

```
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

I servizi pubblicati come Web service possono essere acceduti normalmente utilizzando http e la porta 80. Ciò permette un elevato grado di sicurezza poiché la presenza di firewall, con aperta la porta 80, non impedisce la comunicazione che utilizza SOAP come protocollo.
La grande flessibilità di questo protocollo ha tuttavia un costo: la quantità di bytes necessari per spedire un messaggio SOAP è maggiore di quella necessaria per spedire le stesse informazioni codificate, ad esempio, in binario. Inoltre sia durante la spedizione che il ricevimento del messaggio, è necessario elaborare un documento xml, rispettivamente per la costruzione del messaggio e la seguente lettura delle informazioni. Per dettagli relativi a SOAP si veda W3C.SOAP.

WSDL[19] è un linguaggio basato su XML che permette la descrizione dei Web Services. Permette di specificare il luogo e le operazioni (o metodi) che il servizio mette a disposizione. WSDL descrive i messaggi che sono scambiati da chi fornisce il servizio e da chi lo richiede. I messaggi sono descritti in modo astratto e non collegati ad un particolare formato o meccanismo di trasporto. Un messaggio è composto da una collezione di elementi aventi un tipo. Uno scambio di messaggi è detto *Operation*. Una collezione di operazioni è detta *PortType*. Un *Service* è composto da un insieme di *Ports*, ognuna delle quali è una implementazione di una *PortType*, la quale include tutti i dettagli relativi all'interazione con il servizio. La *PortType* può essere vista come una libreria di funzioni (o un modulo, o una classe) in un linguaggio di programmazione tradizionale. I *message* sono composti da una o più *ports*. Le *ports* possono essere viste come i parametri in una chiamata ad un metodo o a una funzione in un linguaggio di programmazione tradizionale. I *types* definiscono, mediante XML Schema, i tipi dei dati che vengono utilizzati nei messaggi. L'elemento *binding* descrive dettagli relativi al formato e al protocollo utilizzato dai messaggi.

Elemento	Definisce
`<portType>`	Le operazioni realizzate dal servizio
`<message>`	I messaggi usati dal servizio

[19] *WSDL: Web Service Description Language.*

`<types>`	I tipi di dati utilizzati dal servizio
`<binding>`	I protocolli di comunicazione usati dal servizio.

Esempio 6. Struttura di un documento WSDL

```
<definitions>
            <types>
   definition of types........
            </types>

            <message>
   definition of a message....
            </message>

            <portType>
   definition of a port.......
            </portType>

            <binding>
   definition of a binding....
            </binding>
</definitions>
```

Nell'Esempio 6 è possibile vedere un semplice esempio di documento WSDL.

Esempio 7. Documento WSDL che descrive un servizio. Sempre paragonando ad un linguaggio di programmazione comune, l'elemento portType può essere visto come un libreria di funzioni avente nome "glossaryTerms", "getTerm" è una funzione con "getTermRequest" come parametro di input e "getTermResponse" come parametro di ritorno.

```
<message name="getTermRequest">
   <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
   <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
  <operation name="getTerm">
      <input message="getTermRequest"/>
      <output message="getTermResponse"/>
  </operation>
</portType>
```

Il punto di collegamento fra SOAP e WSDL è dato dalla sezione `binding`.
All'esempio 6 e possibile aggiungere la sezione binding per specificare dettagli relativi al protocollo di comunicazione. Vedi Esempio 7. Per dettagli consultare W3C.WSDL.

Esempio 8. Sezione Binding da accodare al documento dell'Esempio X.

```
<binding type="glossaryTerms" name="b1">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
     soapAction="http://example.com/getTerm"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```

Per poter avere un indice organizzato dei Web services disponibili in un ambito non solo locale è possibile utilizzare un sistema di directory per i Web services, chiamato UDDI[20], che permetta di scoprire, descrivere ed integrare i servizi messi a disposizione. UDDI utilizza WSDL per la descrizione dei servizi e comunica utilizzando SOAP. Per dettagli relativi a UDDI si veda W3C.UDDI.

4.2 Microsoft .NET Framework

Nel 2000 Microsoft ha sviluppato un insieme di librerie ed un ambiente di esecuzione chiamato CLI[21]. Tale framework è stato in seguito sottoposto a standardizzazione presso ECMA. Vedi ECMA.CLI.
Il Microsoft .NET Framework, una possibile implementazione ed estensione della CLI, è in pratica nuova piattaforma per lo sviluppo e l'esecuzione di applicazioni locali, applicazioni Web-based e XML Web Services. Tale piattaforma è fisicamente composta da:

- Ambiente di esecuzione per poter gestire codice intermedio non nativo compilabile just in time. Vedi Sezione 4.2.1.
- Insieme di librerie per il supporto dell'applicazione sia in fase di sviluppo (mediante l'utilizzo di funzionalità avanzate fornite dalla varie librerie) sia in fase di esecuzione (mediante l'utilizzo di librerie atte all'interazione con il runtime.).
- Strumenti per lo sviluppo quali compilatori, disassemblatore, debugger etc.
- Estensione del Web server Microsoft IIS 5.0 (IIS 6.0, presente sui Windows 2003 Server, è compatibile in modo nativo con .NET) per il supporto di pagine dinamiche tipo ASP o JSP (Vedi sezione 4.4) eseguite nel nuovo runtime. Tramite tale estensione, è possibile anche pubblicare Web Services che utilizzano gli standards SOAP-XML (Vedi Sezione 4.1)

E' inoltre fornito un nuovo linguaggio di programmazione chiamato C# (da pronunciarsi C sharp. Vedi Sezione 4.3) che insieme a C/C++ e profonde revisioni di Visual Basic e JScript forniscono i principali strumenti di sviluppo.

4.2.1 Common language runtime e librerie di classi.

Il cuore di .NET è la common language runtime (CLR) e un insieme di librerie che forniscono le API per il codice delle applicazioni. La coppia CLR e librerie è basata su concetti comuni anche ad altri ambienti d'esecuzione e sviluppo come la Java Virtual Machine e il Java SDK oppure il runtime visual basic insieme alle sue librerie. Lo stesso sistema operativo, unitamente alle librerie C, può essere visto come ambiente d'esecuzione che fornisce servizi alle applicazioni. L'idea base è quella di fornire livelli d'astrazione per offrire servizi ai livelli superiori e alle applicazioni in modo da facilitare il compito di sviluppatori che possono concentrare le loro risorse sullo sviluppo dell'applicazione e permettere il riutilizzo di codice.

[20] *UDDI: Universal Description, Discovery and Integration.*
[21] *CLI: Common Language Infrastructure.*

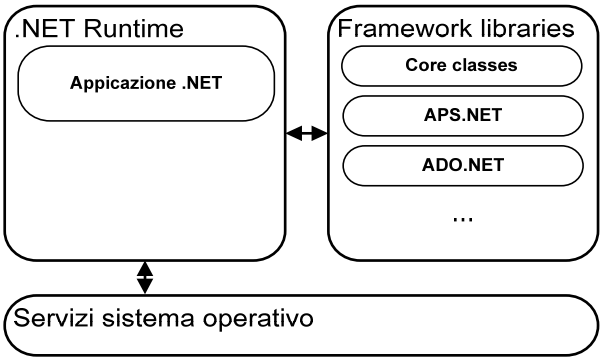


Figura 16: Run-time, sistema operativo e librerie del framework.

Il CLR fornisce servizi come gestione della memoria, dei processi, dei thread e della sicurezza a livello di codice. Inoltre ha un forte ruolo durante la fase di sviluppo fornendo uno strong type system (Vedi Sezione 4.2.2), gestione delle eccezioni cross-language (come si vedrà in seguito è possibile scrivere un'applicazione utilizzando linguaggi di programmazione differenti), dynamic binding etc.
Le librerie del framework forniscono le funzionalità di base come l'input/output, manipolazione di stringhe, gestione della sicurezza, comunicazioni di rete, gestione dei thread, gestione delle interfacce utente etc. Sono presenti inoltre librerie più specifiche come le ADO.NET che permettono un facile accesso ai DBMS, librerie per la manipolazione di documenti XML e librerie ASP.NET (Vedi Sezione 4.4) per la creazione di applicazioni Web-based e l'implementazione di Web Services basati su standards SOAP-XML.
Non essendo l'esplicazione esaustiva di .NET compito di questo documento, saranno riportate alcune caratteristiche fondamentali por poter permettere di capire alcuni dei motivi che hanno fatto scegliere il .NET framework come piattaforma di sviluppo ed esecuzione del progetto lezi.NET.

- **Modello di programmazione consistente**: tutti i servizi del framework sono forniti mediante un comune modello orientato agli oggetti e non come oggi in cui i servizi forniti possono essere acceduti mediante librerie (DLL) e mediante oggetti COM.
- **Modello di programmazione semplificato**: non è più necessario avere a che fare con l'architettura arcana di Win32 e COM. Potranno quindi essere dimenticate entità quali registro di sistema, GUIDs, IUnknown, AddRef, Release, HRESULTS etc. Il framework non astrae questi concetti ma li elimina completamente utilizzando una diversa architettura. La presenza del garbage collector nel runtime semplifica notevolmente la programmazione ad oggetti. La gestione automatica del tempo di vita degli oggetti evita gli errori run-time dovuti a riferimenti che puntano ad oggetti deallocati inavvertitamente dallo sviluppatore prima del necessario.
- **Run once, run always**: quasi tutti gli sviluppatori avranno, almeno una volta, avuto a che fare con problemi legati all'allineamento[22] delle librerie di sistema.

[22] *Per problemi di allineamento delle librerie di sistema si intendono tutti quei problemi di sviluppo, deployment, manutenzione legati alla presenza di versioni diverse della stessa libreria, a librerie diverse eventi lo stesso nome, ad applicazioni che, installando una versione diversa di una libreria, vanno a sovrascrivere la versione esistente precludendo il normale funzionamento di altre applicazioni che ne richiedevano una specifica versione etc.*
E' anche vero che, progettando correttamente una libreria, molti di questi problemi non dovrebbero sussistere, tuttavia la realtà dei fatti è un'altra, realtà piena di revisioni delle interfacce, metodi con side-effect diversi a seconda della versione, etc.

L'architettura .NET risolve definitivamente questi tipi di problemi. E' infatti possibile far si che un'applicazione utilizzi una particolare libreria (un file *.dll) ponendola nella stessa directory dell'eseguibile, oppure, se si vuole rendere disponibile una libreria "system wide", è possibile avere diverse versioni della stessa DLL in un "archivio"[23] di librerie installate nel sistema. Ciò è possibile visto che l'identificatore di una DLL .NET non è composto solo dal nome del file ma è anche da altri metadati (opzionali, ma necessari quando si vuole che una libreria sia pubblicamente utilizzabile o installata nel sistema e utilizzabile da più applicazioni) come ad esempio la chiave pubblica della software house che ha prodotto la DLL oppure la versione/build. In questo modo è garantita un'identificazione univoca.

- **Esecuzione dello stesso codice su diverse piattaforme**: avendo a disposizione un'implementazione del runtime e del framework per diversi sistemi operativi, è possibile eseguire applicazioni .NET anche su macchine non Windows. Esiste, ad esempio, una realizzazione di un framework (runtime + librerie) funzionante per il sistema operativo Linux (Vedi MONO) che è in grado di eseguire codice sviluppato su piattaforme Windows. Il CLR, implementando una macchina virtuale, garantisce pure l'indipendenza del codice dalla particolare piattaforma hardware (x86, non x86, 32 o 64bit etc).
- **Deployment**: fondamentale per la riuscita di un progetto è la facilità (spesso difficoltà) con cui è possibile distribuire un prodotto, renderlo operativo e mantenerlo. Nei sistemi Windows, l'installazione di un'applicazione richiede, oltre che la copia di files anche impostazioni nel registro di sistema, per non parlare del processo di disinstallazione che non è mai in grado di rimuovere completamente tutte le tracce del programma[24]. .NET tenta di risolvere questa situazione poiché è possibile installare e rendere immediatamente operativa un'applicazione semplicemente copiandone la directory che la contiene sul file system. Se necessaria, la configurazione di un'applicazione viene generalmente salvata in file XML predefiniti posizionati sempre nella directory dell'applicazione. Ad esempio per installare lezi.NET che, come si vedrà in seguito, consiste in un applicazione Web a tre livelli, composta da uno o più Web server in front-end, un content management system ed un database, è necessario eseguire una semplice copia di file, impostare alcuni permessi sulle directory del file system, creare le Web application sul Web server e collegare il database al DBMS. Il tutti, se fatto a mano, richiede non più di 15 minuti.
- **Sviluppo rapido**: la grande quantità e qualità delle librerie fornite con il framework permette allo sviluppatore di concentrare le proprie energie sulla business logic della propria applicazione. Ciò, insieme con un modello di programmazione consistente e semplificato, permette uno sviluppo rapido di applicazioni relativamente complesse utilizzando un minor numero di linee di codice per l'implementazione delle stesse funzionalità. Famoso è il confronto fra due differenti realizzazioni della stessa applicazione Web: il Pet Shop (.NET) vs. Per Store (Java). Vedi Figura 17.

[23] *Questo archivio è l'Assemby Cache.*

[24] *Per questi motivi, facendo un parallelo con la termodinamica, definisco Windows, ma anche altri sistemi operativi, come non reversibili in quanto l'entropia, intesa come "disordine" di sistema, può solo aumentare.*

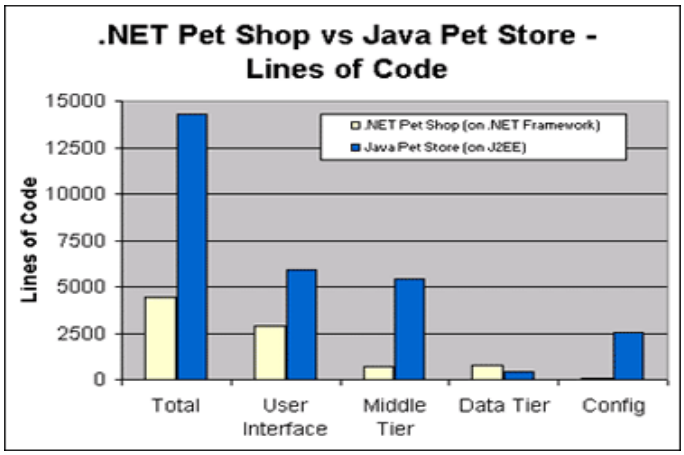


Figura 17: Le due implementazioni forniscono le stesse funzionalità. L'implementazione Java usa J2EE.

Per produrre un'applicazione o un componente con il framework è possibile utilizzare un qualsiasi linguaggio di programmazione scelto fra quelli per cui esiste un compilatore nel framework. Al momento esistono compilatori per C++, Visual Basic, JScript e C#, tuttavia potranno essere facilmente sviluppati, non solo da Microsoft[25], altri compilatori che permettano di produrre codice .NET. Ogni compilatore, tuttavia, non produce direttamente codice eseguibile su una particolare piattaforma hardware. Il codice è invece tradotto in un diverso linguaggio detto Microsoft Intermediate Language (MSIL) che insieme con alcuni metadati diventerà l'eseguibile portabile .NET. Il MSIL è un linguaggio CPU-independent avente la potenza semantica tale da poter supportare tutti (o quasi) i costrutti presenti in linguaggi come C++, Java etc. Codice scritto con un linguaggio semanticamente più semplice, quando compilato, utilizzerà un sottoinsieme di MSIL. I metadati che, insieme al codice MSIL, costituiscono l'eseguibile/libreria contengono informazioni relative al codice stesso (versione, build etc), alla lista di librerie da cui è dipendente e tutta una serie d'informazioni utili al runtime per il caricamento in memoria e la successiva messa in esecuzione. I files prodotti dalla compilazione vengono chiamati "Managed Code" poiché il codice MSIL con cui sono scritti verrà successivamente "trattato" (compilazione just-in-time) ed eseguito. In particolare un managed code che è una libreria è spesso chiamato "Assembly".
Quando un managed code viene lanciato, il runtime, tramite un class loader, carica in memoria il codice MSIL e tutto il codice relativo alle librerie indicate dai metadati dell'eseguibile. Dopo il link fra i vari pezzi di codice il tutto viene compilato in codice macchina per la piattaforma hardware specifica. Vedi Figura 18.

[25] *Questo perché le specifiche del MSIL sono state rese pubbliche e sottoposte a standardizzazione. Esistono produttori di compilatori che hanno come linguaggio pozzo MSIL per linguaggi sorgente come APL, CAML, Cobol, Haskell, Mercury, ML, Oberon, Oz, Pascal, Perl, Python, Scheme, e Smalltalk.*

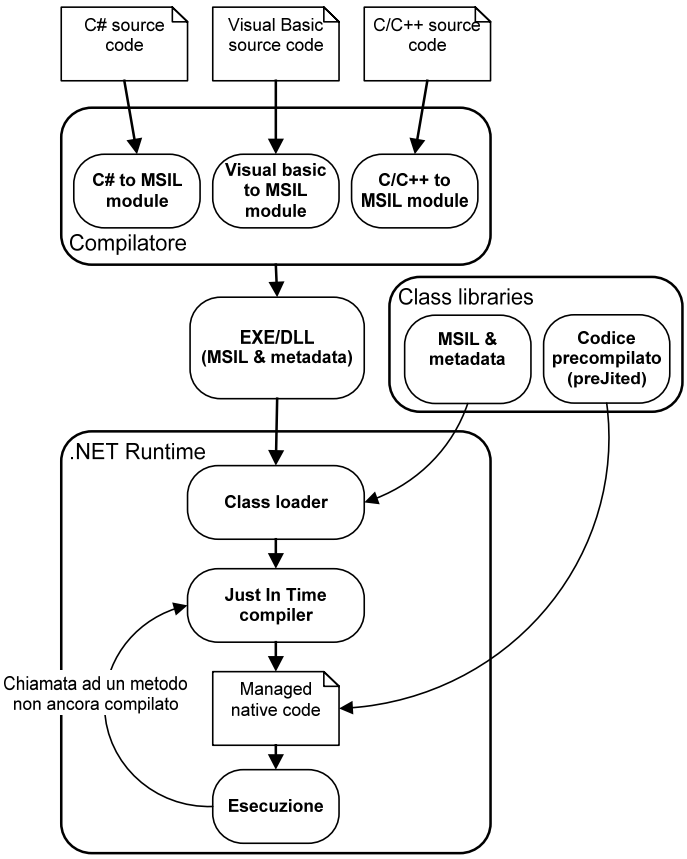


Figura 18: Compilazione ed esecuzione nel runtime .NET

In realtà il codice non è completamente compilato durante la fase d'avvio dell'applicazione poiché tale operazione potrebbe portare a lunghi tempi d'avvio e compilazione di parti di codice che non saranno, in seguito, utilizzate durante l'esecuzione. La compilazione di codice avviene quando un metodo è chiamato per la prima volta. Per le successive, il runtime avrà a disposizione direttamente l'indirizzo del codice macchina in cui il metodo inizia, prodotto durante la prima esecuzione.
Per quanto riguarda le prestazioni, si era abbastanza convinti che tutto l'overhead dovuto al runtime .NET e sopratutto alla compilazione JIT avesse un impatto pesante sulle performance soprattutto a confronto di applicazioni scritte in C/C++ e compilate in codice nativo (Unmanaged code). Invece si è notato come, anche per applicazioni relativamente "pesanti", le prestazioni siano simili a alle corrispettive compilate in codice nativo. Effettivamente il compilatore JIT è in grado, avendo un maggiore numero di informazioni relative all'ambiente di esecuzione rispetto ad un compilatore standard, di ottimizzare la produzione del codice macchina, ad esempio attivando l'utilizzo di registri o particolari set di istruzioni presenti sul processore per cui sta producendo codice. E' in ogni caso vero che codice compilato e ottimizzato per una particolare piattaforma avrà certamente delle prestazioni superiori in quanto potrà essere eseguito direttamente nel runtime del sistema operativo evitando l'ulteriore grado d'astrazione imposto dal runtime .NET.
Esiste, al fine di evitare la compilazioni JIT, anche la possibilità di compilare per una particolare piattaforma alcune librerie o addirittura l'intera applicazione prima dell'esecuzione creando una versione dell'applicazione machine dependent. Ad esempio, le librerie principali del framework, installate nell'assembly cache di sistema, vengono compilate alla prima esecuzione. In seguito il runtime utilizzerà la loro versione già compilata.

L'unico punto dolente, tuttavia anche per il Java Runtime Environment, è il memory footprint dell'applicazione che rispetto a quello della versione nativa risulta circa raddoppiato.

4.2.2 Common type system

Per permettere di far interoperare applicazioni e librerie scritte con linguaggi diversi, ad esempio utilizzare una libreria scritta in Visual Basic all'interno di un'applicazione scritta in C#, è stato necessario formalizzare il type system utilizzato dal runtime. La stessa specifica formale, il Common Type System (CTS) appunto, viene utilizzata dai diversi compilatori che hanno come linguaggio pozzo comune il MSIL. Ad esempio il CTS stabilisce che una classe possa avere zero o più membri:

- **Field**: dato che caratterizza lo stato dell'oggetto. Un field è identificato dal suo nome e dal tipo.
- **Metodo**: funzione che esegue un'operazione sull'oggetto. Modifica generalmente lo stato. I metodi sono identificati da nome, signature[26] e modificatori[27].
- **Proprietà**: per il chiamante sono visti come dei field. Invece per chi implementa sono visti come dei metodi. Le proprietà permettono, infatti, di calcolare il valore da restituire. Le proprietà sono equivalenti all'implementazione dei classici metodi Set/Get per poter leggere/scrivere un valore di un field non direttamente accessibile eventualmente eseguendo delle operazioni. In C# una proprietà ha la seguente forma:

Esempio 9: implementazione di una proprietà di una classe C#.

```
Class Parallelepipedo
{
            private int altezza;
            private int larghezza;
            private int profondità;

            ...

            public int Volume
            {
                  get
                  {
                     int vol = altezza*larghezza*profondità;
                     return vol;
                  }
                  // il metodo Set in questo esempio non è significativo
                  // la proprietà sarà di sola lettura.
                  // set
                  // {
                  // }
            }

}
```

- **Evento**: i membri di tipo event permettono, insieme ai delegate, di implementare un efficace sistema ad eventi fra gli oggetti di un'applicazione.

[26] *La segnature e definita dall'insieme: numero, tipo e sequenza dei parametri e tipo del parametro di ritorno.*
[27] *I modificatori sono attributi che ,ad esempio, danno delle proprietà riguardanti l'accesso al metodi come public, private, protected etc, oppure indicano che tale metodo è un web service.*

Logicamente l'insieme dato dall'unione dei diversi tipi di membri esistenti nei vari linguaggi sorgente deve essere minore e uguale ai tipi di membri esistenti nel CTS ed implementabili mediante MSIL.
Il CTS formalizza tutti gli aspetti relativi a visibilità e accessibilità alle varie entità coinvolte come classi, campi, metodi, definisce tutte le regole per gestire ereditarietà, funzioni virtuali, tempo di vita degli oggetti etc.
L'utilizzo di un CTS è possibile in quanto la stessa classe, con il medesimo comportamento, può essere implementata usando un qualsiasi linguaggio avente una determinata capacità semantica. Questo perché il linguaggio ed il comportamento del codice sono due cose diverse. Certamente la sintassi delle due implementazioni sarà diversa ma il comportamento sarà il medesimo. Il CTS descrive, mediante una sintassi proprietaria, il comportamento della classe. Esistono, tuttavia, casi in cui il CTS e quindi il MSIL non hanno una capacità semantica tale da poter supportare il linguaggio sorgente. Ad esempio il CTS non supporta l'ereditarietà multipla. Se si tenta di creare del managed code da un sorgente C++ che utilizza l'ereditarietà multipla si incorrerà in un errore in fase di compilazione.
Per maggiori informazioni sul .NET Framework vedi MS.NET.FRAMEWORK

4.3 C#

E' un linguaggio di programmazione OO[28]. Oltre che alla standardizzazione della CLI, ECMA si occupa anche della standardizzazione di C# (Vedi ECMA.C#). Sebbene l'implementazione di Microsoft di C# si appoggi sulle librerie e sul runtime di CLI, altre possibili implementazioni possono appoggiarsi ad ambienti di esecuzione diversi.
I principali obiettivi definiti durante lo sviluppo del linguaggio sono:

- C# deve essere semplice e orientato agli oggetti.
- Il linguaggio deve soddisfare principi di ingegneria del software quali: strong type checking, array bounds checking, rilevamento di utilizzo di variabili non inizializzate e garbage collection automatico.
- Il linguaggio deve supportare lo sviluppo di componenti utilizzabili in ambienti distribuiti.
- La portabilità del codice e dei programmatori è un aspetto importante soprattutto per i programmatori cha hanno familiarità con altri linguaggi come C,C++ e Java.
- Sebbene le applicazioni sviluppate con C# debbano utilizzare in modo efficiente risorse quali processore e memoria, il linguaggio non è pensato per competere con C o assembler.

Di seguito è presentato un esempio della classica prima applicazione che di solito si sviluppa durante la fase di apprendimento di un nuovo linguaggio.

Esempio 10. Il classico programma Hello, World.

```
using System;
class Hello
{
            static void Main() {
                  Console.WriteLine("hello, world");
            }
}
```

[28] *OO: Object oriented.*

Il codice sorgente per un programma scritto in C# è di solito memorizzato in un file di testo avente estensione .cs. Per compilare il precedente codice è possibile utilizzare un compilatore a riga di comando.

```
csc hello.cs
```

La compilazione produce un eseguibile avente nome hello.exe che mandato in esecuzione darà origine al seguente output:

```
hello, world
```

Esaminando il codice è possibile notare che:

La direttiva `using` permette di specificare che il codice utilizza una libreria di classi il cui namespace è System che è fornita dalla CLI. L'utilizzo delle direttive all'interno del codice permette di passare dalla notazione `System.Console.WriteLine` alla notazione `Console.WriteLine`.
Il punto d'entrata di una applicazione è sempre un metodo statico di nome `Main`.
L'output è prodotto utilizzando una libreria di classi.
L'applicazione non utilizza gli operatori "::" e neppure "->" ma soltanto la dot-notation. L'operatore "->" può essere utilizzato solo per piccole porzioni di codice marcato come "unsafe".
Il programma non utilizza `#include` per importare codice in altri files. Le dipendenze sono gestite in modo simbolico invece che testuale. In questo modo è possibile scrivere un programma utilizzando diversi linguaggi di programmazione. Ad esempio, la classe Console potrebbe essere stata scritta utilizzando Visual Basic .NET invece di C#.
Questo testo non ha comunque come scopo quello di parlare approfonditamente del linguaggio. Per maggiori dettagli vedere ECMA.C#.

4.4 Pagine Web server-side ASP .NET

La pagina attive ASP.NET permettono la costruzione di una pagina html ad ogni richiesta del client nello stesso modo in cui accade per APS, JSP o PHP. Le pagine ASP.NET sono essenzialmente un documento XML, con estensione .aspx, in cui sono presenti tag standard html e tag speciali aventi namespace "asp". Il codice della pagina è racchiuso dal tag <% %> in cui è presente la logica per la costruzione dinamica della pagina. Per permettere una separazione della business logic dal lato di presentazione è possibile utilizzare un ulteriore file di testo, avente estensione .cs o .vb a seconda del linguaggio con cui è stato scritto, in cui è contenuto tutto il codice.
Questa tecnica, chiamata codebehind, permette di ottenere una migliore organizzazione nell'applicazione. In ogni pagina di codice è dichiarata una classe. I metodi e le proprietà di tale classe forniscono supporto per la corrispondente pagina aspx. Quando al Web server .NET arriva la richiesta per una pagina aspx. questi istanzierà una classe che eredita dalla classe scritta nel codebehind in modo da poter avere accesso ai suoi metodi e proprietà. Durante il parsing della pagina aspx i tag html vengono ignorati mentre i tag asp sono sostituiti dinamicamente da html che rispecchi lo stato elaborato nelle sezioni di codice della pagina aspx oppure nel codebehind. Il codice html di output prodotto tiene conto anche del tipo di browser del client in modo da aumentare la compatibilità con i browser obsoleti e sfruttare le caratteristiche di quelli più moderni senza, tuttavia dover modificare il codice dell'applicazione.

Esempio 11. Codice di una pagina aspx contenente tre componenti asp: un asp:TexBox, un asp:button e un asp:Label. UQesti tag verrannosostituiti al momento dell'elaborazione con hatml semplice. Il codebehind necessario per utilizzare "programmatically" tali componenti è visibile nell'esempio seguente.

```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs" AutoEventWireup="false"
Inherits="WebApplication1.WebForm1" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
		<HEAD>
			<title>WebForm1</title>
			<meta name="vs_showGrid" content="False">
			<meta name="GENERATOR" Content="Microsoft Visual Studio 7.0">
			<meta name="CODE_LANGUAGE" Content="C#">
			<meta name="vs_defaultClientScript" content="JavaScript">
			<meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
		</HEAD>
		<body>
			<form id="Form1" method="post" runat="server">
			<P> </P>
			<P>
				<asp:TextBox id="TextBox1"
runat="server"></asp:TextBox></P>
			<P>
				<asp:Button id="Button1" runat="server"
Text="Button"></asp:Button></P>
			<P>
				<asp:Label id="Label1"
runat="server">Label</asp:Label></P>
			</form>
		</body>
</HTML>
```

Esempio 12: Codice C# relativo alla pagina aspx dell'esempio precedente. Si noti che vengono dichiarati gli handler per gli oggetti corrispondenti ai componenti con namespace asp nella pagina aspx. Interessante, per una pagina Web, è anche la gestione degli eventi che permette ad esempio di chiamare il metodo Page_Load quando inizia l'elaborazione della pagina.

```
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace WebApplication1
{
		/// <summary>
		/// Summary description for WebForm1.
		/// </summary>
		public class WebForm1 : System.Web.UI.Page
		{
			protected System.Web.UI.WebControls.TextBox TextBox1;
			protected System.Web.UI.WebControls.Button Button1;
			protected System.Web.UI.WebControls.Label Label1;

			private void Page_Load(object sender, System.EventArgs e)
			{
				// Put user code to initialize the page here
			}

			#region Web Form Designer generated code
			override protected void OnInit(EventArgs e)
```

```
                    {
                          //
                          // CODEGEN: This call is required by the ASP.NET Web
Form Designer.
                          //
                          InitializeComponent();
                          base.OnInit(e);
                    }

                    /// <summary>
                    /// Required method for Designer support - do not modify
                    /// the contents of this method with the code editor.
                    /// </summary>
                    private void InitializeComponent()
                    {
                          this.Load += new System.EventHandler(this.Page_Load);

                    }
                    #endregion
              }
}
```

Una caratteristica fondamentale della pagina ASP.NET è il supporto automatico per lo stato della singola pagina. Quando si scrive il codice per una pagina si ha a disposizione una “statebag” in cui tutti i componenti della pagina memorizzano il loro stato. Tale stato sarà recuperato alla successiva elaborazione della pagina. Ciò è implementato mediante una tag input nascosto in cui è memorizzata una serializzazione criptata di tutti gli oggetti memorizzati nella statebag. Tale stringa sarà riletta al successivo post e permetterà ad ogni componente di recuperare il proprio stato. Questo tecnica, chiamata roundtrip, permette di rendere statefull l’html.

Esempio 13. Nella parte di pagina di questo esempio è possibile vedere il campo __VIEWSTATE in cui i diversi componenti memorizzano il proprio stato in modo da poterlo recuperare alla successiva operazione di post. Il contenuto del campo nascosto, facendo parte di una form, viene rispedito al server nella request successiva. In questo modo è possibile mantenere lo stato durante le varie richieste della stessa pagina. Visivamente è possibile avere l’impressione di essere in presenza di una applicazione e non di una pagina Web.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
              <HEAD>
                    <title>Course</title>
                    <meta content="Microsoft Visual Studio 7.0" name="GENERATOR">
                    <LINK href="wxp.css" type="text/css" rel="stylesheet">
              </HEAD>
              <BODY leftMargin="4" topMargin="2">
                    <meta content="C#" name="CODE_LANGUAGE">
                    <meta content="JavaScript" name="vs_defaultClientScript">
                    <meta content="http://schemas.microsoft.com/intellisense/ie5"
name="vs_targetSchema">
                    <form name="Course" method="post" action="Start.aspx"
id="Course">

<input type="hidden" name="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" value="" />
<input type="hidden" name="__VIEWSTATE"
value="dDw4MDQ5MTE0MzQ7dDw7bDxpPDE+Oz47bDx0PDtsPGk8Mz47aTw1PjtpPDk+Oz47bDx0PDtsPG
k8MT47aTwzPjtpPDU+O2k8Nz47aTw5Pjs+O2w8dDxwPHA8bDxWaXNpYmxlOz47bDxvPHQ+Oz4+Oz47bDx
pPDE+Oz47bDx0PHA8cDxsPFRleHQ7PjtsPFdlYlRhbGsgQ3ViZTs+Pjs+Ozs+Oz4+Pjs+Ozs+O3Q8O2w8
aTwxPjs+O2w8dDxwPHA8bDxUZXh0Oz47bDxQdWJibGljYXRvIGlsIGNvcnNvIGRpIFdlYiBEZXNpZ24gZ
GVsIFByb2YuIFBhb2xpbmkuClNvbm8gcHJlc2VudGkgbGUgcHJpbWEgZHVlIGxlemlvbmkuOz4+Oz47Oz
47Pj47dDxwPHA8bDxUZXh0Oz47bDxCcnVuYSBTdGVmYW5vOz4+Oz47Oz47dDxwPHA8bDxUZXh0Oz47bDw
yMzs+Pjs+Ozs+O3Q8cDxwPGw8VGV4dDs+O2w8MTs+Pjs+Ozs+Oz4+Oz4+Oz4+O3Q8cDxwPGw8Vmlz
```

```
aWJsZTs+O2w8bzxmPjs+Pjs+O2w8aTwyPjtpPDk+Oz47bDx0PEA8X3NlbGY7X3NlbGY7IzsjOz47Oz47d
DxwPHA8bDxWaXNpYmxlOz47bDxvPGY+Oz4+Oz47Oz47Pj47Pj47Pj47bDxtZW51QmFyQnV0dG9uOz4+HY
LRUxeBlDUzRniw1EJ1WvhDov0=" />


<script language="javascript">
<!--
              function __doPostBack(eventTarget, eventArgument) {
                    var theform = document.Course;
                    theform.__EVENTTARGET.value = eventTarget;
                    theform.__EVENTARGUMENT.value = eventArgument;
                    theform.submit();
              }
// -->
</script>
```

Parte 2 : Lavoro svolto

5 Requisiti

lezi.NET deriva idealmente dall'applicazione WMV lezi II[29] precedentemente sviluppato per un progetto di informatica grafica. WMV Lezi II è un'applicazione Web scritta in ASP che permette la pubblicazione di contenuti di learning. L'unita fondamentale è il corso suddiviso in lezioni a loro volta suddivise in parti. Ogni parte era composta da una collezione di filmati o tracce audio, da una collezione di diapositive e da un insieme di documenti scaricabili e hyperlink relativi al contesto della parte. L'applicazione WMV Lezi II è quindi fondamentalmente composta da: interfaccia di gestione dei corsi, interfaccia per la creazione di un corso, interfaccia per la fruizione di un corso e interfaccia per la gestione dell'utenza, dei gruppi e dei permessi. Era quindi naturale soddisfare i requisiti WMV Lezi II estendendone le caratteristiche e migliorandone alcuni aspetti. Lezi.NET doveva quindi essere un'applicazione Web principalmente in grado di fornire le funzionalità già esposte nella Sezione 1 di introduzione che vengono per comodità qui riportate:

- **Ambiente di fruizione Web-based**: l'utente tramite il proprio browser deve poter fruire i contenuti di un corso in modo semplice ed automatico mediante un'interfaccia coerente e usabile.
- **Ambiente di gestione dei corsi Web-based**: gli utenti della piattaforma, in base ai loro diritti, devono avere una visione personalizzata della lista dei corsi presenti nel sistema.
- **Ambiente di editing dei corsi Web-based**: gli utenti appartenenti al gruppo di "creatori di corsi" devono poter creare lezioni utilizzando delle risorse in precedenza archiviate nel sistema o caricabili anche durante la fase di produzione.
- **Ambiente di gestione delle risorse Web-based**: deve essere possibile caricare nel sistema le risorse (filmati, diapositive, documenti etc) che potranno essere utilizzate in fase di editing dei corsi in modo da avere un archivio personale, eventualmente condiviso, del proprio materiale didattico.
- **Ambiente di gestione delle iscrizioni ai corsi Web-based**: per non obbligare l'utente creatore di corsi a distribuire manualmente i diritti agli utenti fruitori per un suo corso, deve essere disponibile una procedura guidata che faciliti tale compito.
- **Ambiente di gestione degli utenti e dei gruppi Web-based**: deve essere disponibile uno strumento che permetta gestire utenti e gruppi di utenti per poter supportare processi di autenticazione e autorizzazione necessari per poter eseguire l'accesso al sistema e por poter applicare una logica di distribuzione dei diritti basta su liste di controllo dell'accesso.

[29] *L'autore è Stefano Bruna (n.d.r.)*

6 Progettazione

6.1 Ruoli, gruppi e utenti

Per gestire un sistema complesso come un LMS è necessaria una gestione avanzata dei ruoli[30]: un ruolo per gli utenti, un ruolo per gli utenti creatori di contenuti ed un ruolo per gli amministratori dell'applicazione. Per poter associare un ruolo ad un utente è stato implementato un modello role-based basato su gruppi e utenti simile a quello che si può trovare in molti sistemi operativi multi utente. Il sistema fornisce quattro gruppi predefiniti che definiscono i ruoli base:

- **Everyone**: gruppo a cui appartengono tutti gli utenti che interagiscono con il sistema (anche gli utenti guests).
- **Users**: gruppo che contiene gli utenti del sistema registrati. Tale gruppo estende le caratteristiche del gruppo Everyone.
- **Creators**: gli utenti appartenenti a questo gruppo hanno a disposizione funzionalità quali la gestione dei corsi creati, la gestione delle risorse utilizzabili per la creazione dei corsi, la gestione dei propri gruppi ed utenti e distribuzione diritti, gestione news e gestione dei progetti per la creazione di corsi. Tale gruppo estende le caratteristiche del gruppo Users.
- **Administrators**: gli utenti appartenenti a questo gruppo hanno la possibilità di gestire la configurazione del sistema e di modificare, pur non avendo diritti espliciti, le access control list (ACL) degli oggetti quali corsi, risorse progetti etc. Tale gruppo estende le caratteristiche del gruppo Creators.

Il ruolo che utente ricopre è determinato dinamicamente in base all'appartenenza diretta e indiretta ai gruppi predefiniti. Un utente che appartiene direttamente o indirettamente a più gruppi avrà un ruolo complessivo basato sulla somma dei diritti dei gruppi d'appartenenza.

6.2 Distribuzione dei diritti: ACL e policies

La distribuzione dei diritti viene eseguita mediante access control list e policies di sistema. Per le principali entità di sistema quali corsi, risorse e progetti utilizzano le ACL per garantire maggiore flessibilità e potenza nell'assegnazione di diritti agli oggetti, mentre per le entità più semplici, ad esempio un pannello dell'interfaccia, e per le procedure, come la possibilità di creare e gestire utenti, si adoperano policies.
Le ACL associano un oggetto ad un ruolo ricoperto da un utente garantendo all'utente i diritti indicati dall'ACL stessa. Ad esempio, ad un corso è possibile associare una ACL che garantisce diritti di lettura per gruppo e diritti di modifica e lettura per un utente. Gli appartenenti ad un gruppo ereditano i diritti del gruppo a cui appartengono. Vedi Figura

[30] Con ruolo si intende la possibilità da parte di un utente di eseguire determinate operazioni caratteristiche di una o più categorie predefinite a cui l'utente appartiene direttamente o indirettamente. Ad esempio, se l'utente appartiene al gruppo predefinito Creators avrà la possibilità di accedere alle sezioni dell'applicazione in cui è possibile creare un corso. L'appartenenza indiretta si ha,ad esempio, quando un utente appartiene ad un gruppo che sua volta appartiene ad un gruppo predefinito.

19. Per garantire ulteriore flessibilità, il modello utilizzato permette l'annidamento di gruppi.

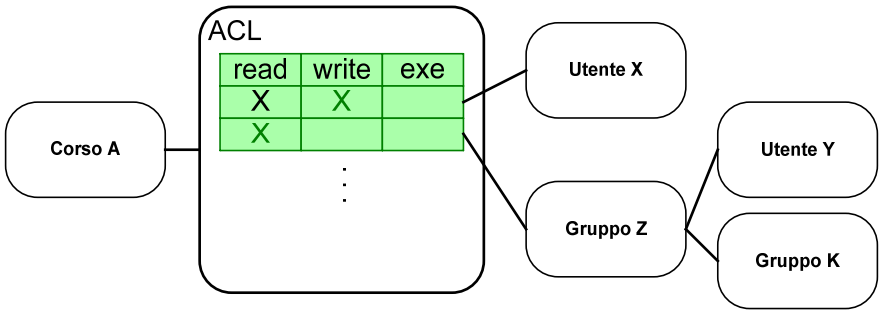


Figura 19: Distribuzione dei diritti mediante ACL ad un corso.

La Figura 19 deve essere così interpretata: L'utente X ha diritti di lettura (read) e scrittura (write) sul corso A. Gli appartenenti al gruppo Z (Utente Y e utente K) hanno diritti di sola lettura sul corso A.
Per poter determinare i diritti effettivi per un utente su un oggetto è necessario:

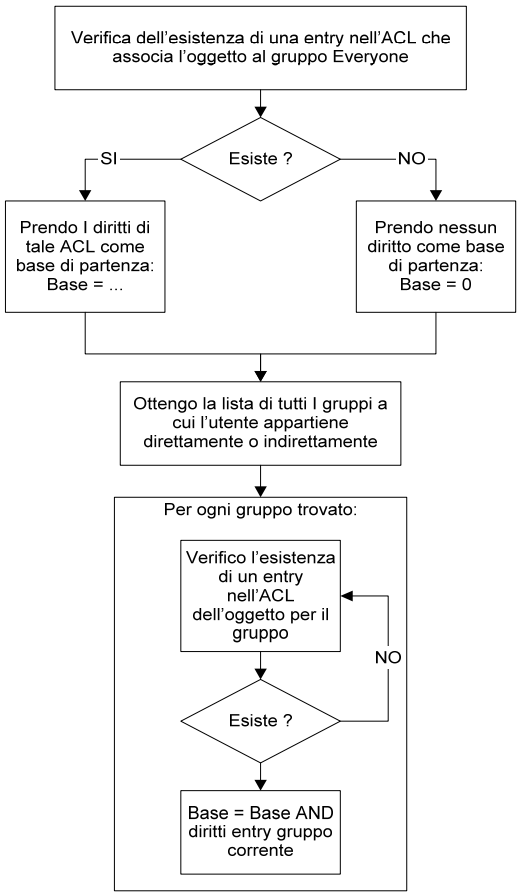


Figura 20: Algoritmo per determinare i diritti effettivi di un utente su un oggetto. Questo algoritmo è stato effettivamente utilizzato nell'implemtazione.

Le policies permettono, invece, di definire un comportamento del sistema associato al particolare ruolo che l'utente ricopre. Alcune possibili policies potrebbero essere quelle di Figura 20.

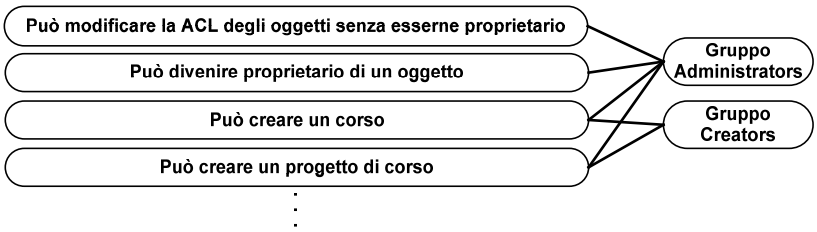


Figura 21: Esempi di policies.

Le prima policy di Figura 21 deve essere così interpretata: Un qualsiasi utente appartenente al gruppo Administrators può modificare la ACL degli oggetti senza esserne proprietario[31]. Nel modello role-based utilizzato sono presenti inoltre alcuni principi base universali:

- Il proprietario di un oggetto può sempre modificarne la ACL.
- Tutti gli utenti appartengono automaticamente al gruppo everyone.
- Neppure gli amministratori possono aggirare il meccanismo delle ACL. Per poter, ad esempio, modificare un oggetto un utente deve prima modificare la ACL, impostare i diritti di scrittura per il suo utente o per il gruppo Administrators ed effettuare infine le modifiche.

6.3 Funzionalità

I requisiti impongono la realizzazione delle funzionalità schematizzate nei casi d'uso di Figura 22:

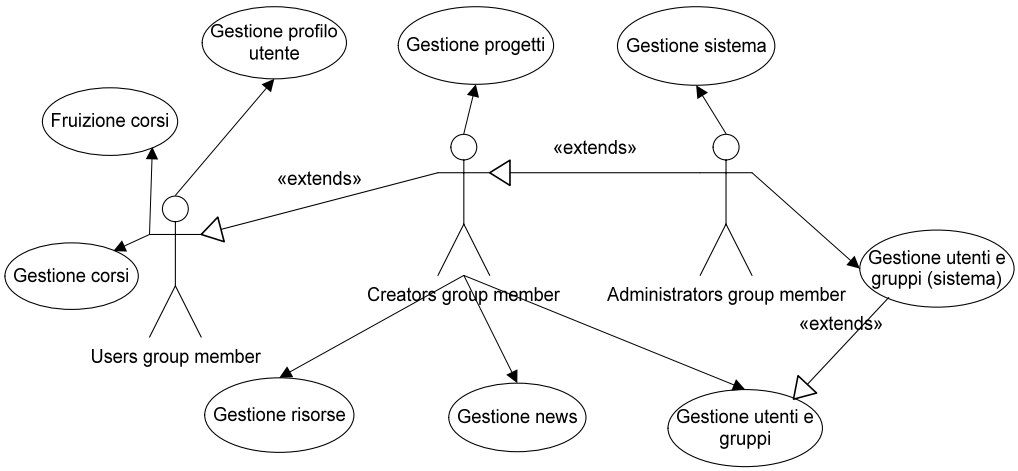


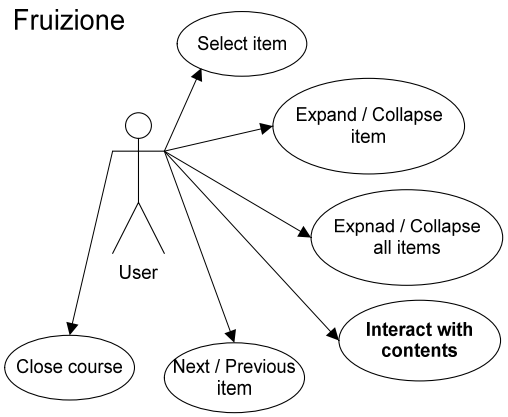
Figura 22: Casi d'uso che sintetizzano le funzionalità da realizzare.

Vengono di seguito presentati altri casi d'uso che hanno come scopo quello di mostrare in dettaglio quali sono le funzionalità che il sistema deve fornire per soddisfare i requisiti richiesti. Tali diagrammi e le relative descrizioni testuali danno una descrizione piuttosto dettagliata[32] delle varie operazioni descritte in Figura 22. Nella sezione 7 ed in particolare nella 8 saranno chiariti tutti i dettagli e le problematiche che sorgono durane la fase di

[31] *Questa policy potrebbe essere sostituita da: Può diventare proprietario dell'oggetto. Naturalmente il proprietario dovrà avere accesso completo all'oggetto quindi anche alla modifica della sua ACL.*
[32] *E piuttosto noiosa da leggere.*

implementazione. Nelle tabelle descrittive dei casi d'uso seguenti con il termine "sistema" si intende l'LMS generico che poi verrà implementato, quando invece si utilizza, ad esempio, il termine "servizio di distribuzione" ci si riferisce al servizio di distribuzione (delivery service) presente nell'architettura del LMS di Figura 1. Nella successiva fase d'implementazioni tali servizi saranno mappati su applicazioni e server.

- **Fruizione corso**: interfaccia Web-based che permette la fruizione di un corso IMS–SCORM. E' presente un frame da utilizzare per la navigazione nell'albero dei contenuti. Il frame centrale visualizza i contenuti del corso.



Le seguenti funzionalità saranno implementate mediante l'ambiente d'esecuzione lezi.NET viewer.

Use case	Select item
Precondizione	Essere nell'ambiente di esecuzione.
Percorso base	1. L'utente seleziona dalla struttura ad albero una voce del table of contents. 2. Il sistema visualizza nel frame centrale il contenuto relativo.
Percorso alternativo	
Postcondizione	Un nuovo item è visualizzato nel frame dei contenuti.
Percorso d'eccezione	
Altri	

Use case	Expand / Collapse item
Precondizione	Essere nell'ambiente di esecuzione.
Percorso base	1. L'utente espande un ramo dell'albero dei contenuti. 2. Il sistema ridisegna l'albero con il ramo corrispondente espanso. Non modifica la selezione corrente.
Percorso alternativo	
Postcondizione	Un nuovo item è visualizzato nel frame dei contenuti.
Percorso d'eccezione	
Altri	

Use case	Interact with contents
Precondizione	Essere nell'ambiente di esecuzione.
Percorso base	1. L'utente interagisce con i contenuti mediante l'interfaccia proposta dal contenuto stesso. Ad esempio, se il contenuto propone l'esecuzione di un test con domande a risposta multipla l'utente risponderà selezionando i radio button delle risposte. 2. Il sistema, che fornisce il run-time per il contenuto, garantisce l'accesso alla struttura dati CMI per la lettura e scrittura delle informazioni relative al contento in esecuzione.
Percorso alternativo	
Altri	

Use case	Next / Previous item
Precondizione	Essere nell'ambiente di esecuzione.
Percorso base	1. L'utente, premendo sul pulsante next/previous sposta sa selezione dal contenuto corrente al prossimo/precedente contenuto determinato in base ad una linearizzzazione dell'albero. 2. Il sistema sposta la selezione, interrompendo l'esecuzione del contenuto corrente, e lancia l'esecuzione del prossimo/precedente contenuto.
Percorso alternativo	2. Se si nel primo e nell'ultimo item la selezione corrente non viene modificata.
Altri	

- **Gestione corso**: pagina composta da una lista di proprietà dettagliate del corso, una serie di azioni da eseguire relative al corso e la ACL del corso. Prima di procedere con la descrizione dei casi d'uso relativi alla gestione dei corsi necessario definire il modello su cui si basano i casi d'uso seguenti. Un corso può trovarsi in tre stati differenti:

- **Registrato**: registrato nel sistema, quindi presente nella lista dei corsi disponibili.

- **Deployed**: quindi disponibile tramite il servizio di distribuzione per gli utenti aventi i ruoli di Administrators e Creators

- **Published**: quindi disponibile tramite il servizio di distribuzione a tutti gli utenti aventi un qualsiasi ruolo.

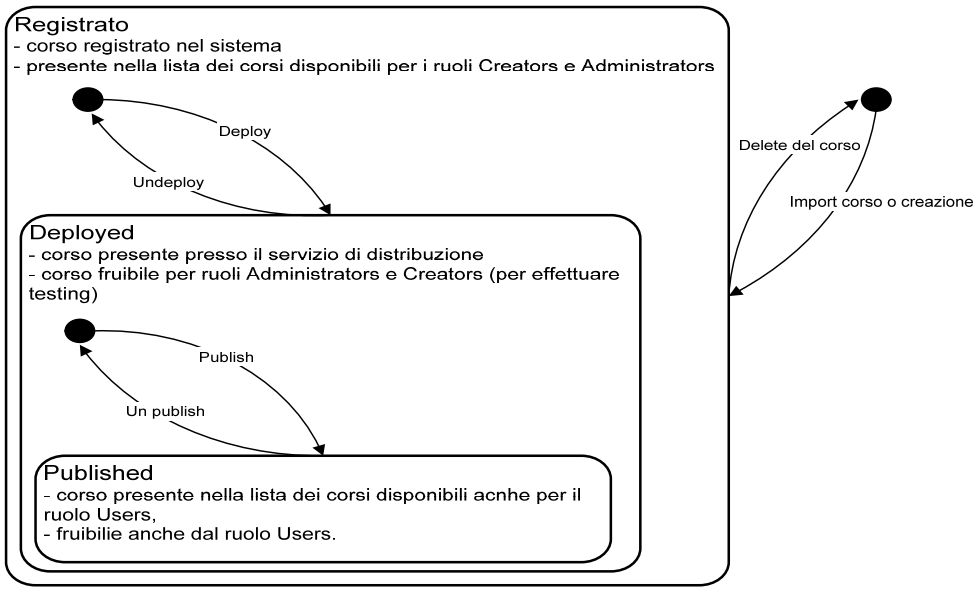
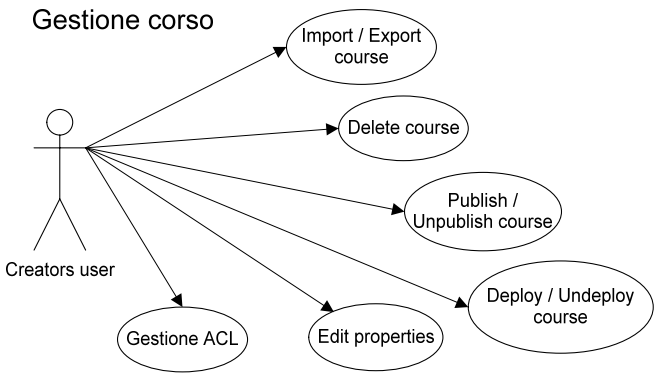


Figura 23: Stati di un corso.

La possibilità di fruire di un corso è dipendente anche dai diritti che l'utente ha sul tal corso. Ad esempio: anche se un corso risulta essere pubblicato solo gli utenti aventi diritti sufficienti per fruirlo potranno eseguirlo. Riassumendo gli stati Esistente, Deployed, Published definiscono la disponibilità del corso per gli utenti in funzione del loro ruolo, mentre le ACL definiscono l'autorizzazione all'esecuzione delle operazioni sul corso. Vedi Figura 23.



Use case	Import course
Precondizione	- Essere nella pagina di gestione del corso - Avere diritti di scrittura sul corso.
Percorso base	1a. L'utente, premendo il tasto Import può, dopo aver compilato un form per l'inserimento di alcuni metadati, inserire nel sistema un package IMS/SCORM. 1b. L'utente, premendo il tasto export, richiede il download di un package IMS/SCORM di un corso. 2a. Il sistema esegue l'upload del package e lo registra fra i corsi disponibili. 2b. Il sistema esegue il download del package.
Percorso alternativo	

Postcondizione	a: Un nuovo corso è presente nella lista dei corsi disponibili b: Nessuna modifica dello stato del sistema.
Percorso d'eccezione	2. Se si verifica un qualche tipo di errore durante la fase di importazione del package il sistema esegue un rollback e si riporta nello stato precedente la richiesta.
Altri	

Use case	Deploy / Undeploy del corso
Precondizione	- Essere nella pagina di gestione del corso - Avere diritti di scrittura sul corso.
Percorso base	1. L'utente, premendo Deploy decide di rendere fruibile il corso agli utenti appartenenti al gruppo Creator e Administrators. I diritti per la fruizione, impostati tramite ACL devono essere comunque impostati. Il deploy è una fase precedente alla vera e propria pubblicazione ed è utilizzata normalmente dall'autore del corso per verificare il corretto funzionamento del corso nell'ambiente di esecuzione vero e proprio. (Run-time SCORM). E' necessario specificare il server di distribuzione su cui effettuare l'operazione. L'Undeploy esegue l'operazione inversa. 2. Il sistema esegue il deployment del package sul server di distribuzione selezionato.
Percorso alternativo	
Postcondizione	Il corso e presente sul server di distribuzione dei contenuti
Percorso d'eccezione	2. Se si verifica un qualche tipo di errore durante la fase di deployment il sistema esegue un rollback e si riporta nello stato precedente la richiesta.
Altri	

Use case	Publish / Unpublish del corso
Precondizione	- Essere nella pagina di gestione del corso - Avere diritti di scrittura sul corso.
Percorso base	1. L'utente, premendo Publish rende disponibile il corso, di cui si è eseguito precedentemente il Deploy, per la fruizione da parte di tutti gli utenti aventi diritti sufficienti. L'Unpublish esegue l'operazione inversa. 2. Il sistema imposta lo stato del corso da Deployed a Published.
Percorso alternativo	
Postcondizione	Il corso è pubblicato.
Percorso d'eccezione	2. Se si verifica un qualche tipo di errore durante la fase di pubblicazione il sistema esegue un rollback e si riporta nello stato precedente la richiesta.
Altri	

Use case	Delete del corso
Precondizione	Essere nella pagina di gestione del corso, il corso non deve essere Deployed ed avere diritti di scrittura sul corso.
Percorso base	1 L'utente cancella un corso presente nel sistema.

	2. Il sistema cancella il corso dalla lista dei corsi disponibili.
Percorso alternativo	
Postcondizione	Il corso non è più elencato fra i corsi disponibili.
Percorso d'eccezione	2. Se si verifica un qualche tipo di errore durante la fase di cancellazione il sistema esegue un rollback e si riporta nello stato precedente la richiesta.
Altri	

Use case	Edit properties
Precondizione	- Essere nella pagina di gestione del corso - Avere diritti di scrittura sul corso.
Percorso base	1. L'utente modifica il valore di alcuni metadati del corso, come ad esempio l'abstract del corso. 2. Il sistema aggiorna il corso con i nuovi valori.
Percorso alternativo	
Postcondizione	Le proprietà modificate contengono i nuovi valori.
Percorso d'eccezione	
Altri	

Use case	Gestione ACL
Precondizione	Essere nella pagina di gestione del corso, se il ruolo maggiore a cui si appartiene è Creators sono necessari anche i diritti di scrittura sul corso. Se si appartiene al ruolo Administrators è sempre comunque possibile modificare le ACL.
Percorso base	1. L'utente modifica il valore di alcuni metadati del corso, come ad esempio l'abstract del corso. 2. Il sistema aggiorna il corso con i nuovi valori.
Percorso alternativo	
Postcondizione	Le proprietà modificate contengono i nuovi valori.
Percorso d'eccezione	
Altri	

- **Gestione progetti**: per poter creare un corso mediante l'editor interno è necessario creare un progetto di corso. E' possibile, in seguito, editare il progetto del corso, salvarlo ed infine produrre il corso mediante l'operazione di build. Lo stato del progetto passa attraverso la macchina a stati in Figura 24.

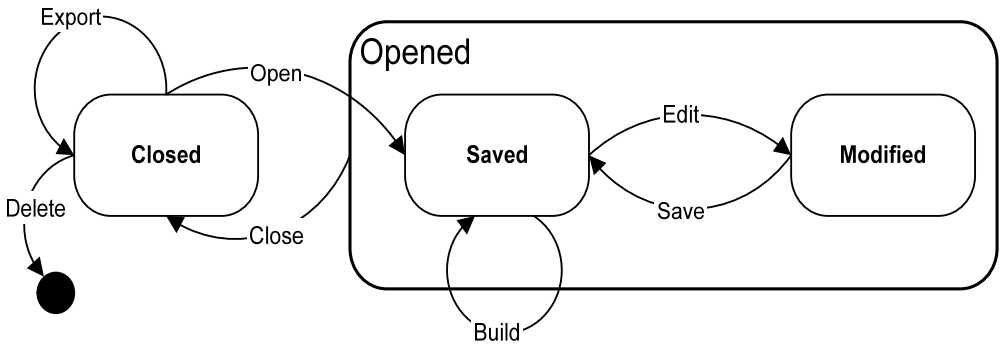
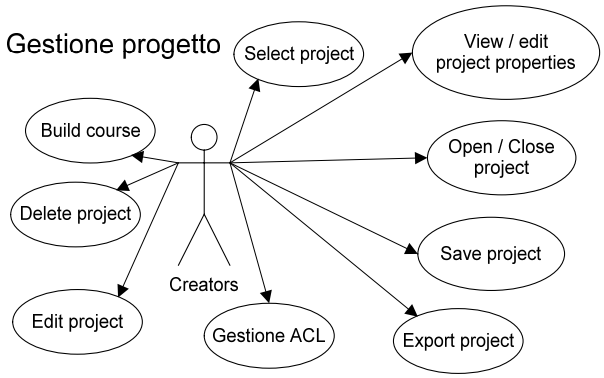


Figura 24: stati di un progetto



Use case	Select project
Precondizione	Essere nella pagina di gestione dei progetti.
Percorso base	1. L'utente seleziona il progetto dalla lista di progetti disponibili. 2. Il sistema seleziona il progetto scelto visualizzandone le proprietà.
Percorso alternativo	
Postcondizione	Viene visualizzata la pagina di gestione del progetto.
Percorso d'eccezione	
Altri	

Use case	View / edit properties
Precondizione	Essere nella pagina di gestione del progetto.
Percorso base	1. L'utente modifica le proprietà del corso come, ad esempio, la descrizione del progetto, l'identificatore della default organization usata (Vedi sezione 3.1.2) oppure il titolo e l'abstract che verranno usati per la creazione del corso. (Operazione di build). 2. Il sistema aggiorna il progetto con i nuovi valori.
Percorso alternativo	
Postcondizione	Vengono visualizzato i nuovi valori.
Percorso d'eccezione	
Altri	

Use case	Open project
Precondizione	- Essere nella pagina di gestione del progetto - Avere i diritti di scrittura sul progetto e il progetto deve essere nello stato Closed.
Percorso base	1. L'utente apre il progetto per poterlo editare. 2. Il sistema apre il progetto e permette all'utente di effettuare delle modifiche. Il sistema, inoltre, blocca l'accesso al progetto da parte di altri utenti. Tale accesso verrà ripristinato alla chiusura del progetto da parte dell'utente corrente.
Percorso alternativo	
Postcondizione	Il progetto è nello stato Opened – Saved.
Percorso d'eccezione	
Altri	

Use case	Close project
Precondizione	- Essere nella pagina di gestione del progetto - Avere i diritti di scrittura sul progetto - Il progetto deve essere nello stato Opened - Il particolare utente deve aver in precedenza aperto il progetto.
Percorso base	1. L'utente chiude il progetto. 2. Il sistema chiude il progetto.
Percorso alternativo	
Postcondizione	Il progetto è nello stato Closed.
Percorso d'eccezione	
Altri	

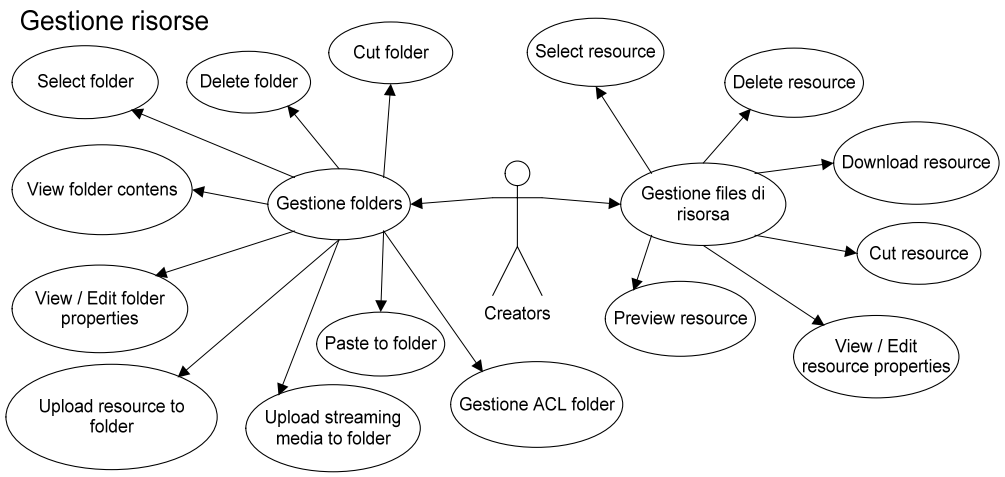
Use case	Save project
Precondizione	- Essere nella pagina di gestione del progetto - Avere i diritti di scrittura sul progetto - Il progetto deve essere nello stato Opened - Il particolare utente deve aver in precedenza aperto il progetto.
Percorso base	1. L'utente, effettuate le modifiche al progetto, salva il progetto rendendole persistenti. E' necessario eseguire tale operazione prima di effettuare il build del progetto per creare il corso. 2. Il sistema salva in modo permanente il lavoro dell'utente sovrascrivendo il vecchio salvataggio.
Percorso alternativo	
Postcondizione	Il progetto è nello stato Saved.
Percorso d'eccezione	
Altri	

Use case	Build course
Precondizione	- Essere nella pagina di gestione del progetto

	- Avere i diritti di scrittura sul progetto - Il progetto deve essere nello stato Saved - Il particolare utente deve aver in precedenza aperto il progetto.
Percorso base	1. Utilizzando come sorgente il progetto corrente, l'utente crea un corso che andrà ad aggiungersi alla lista di corsi disponibili. 2. Il sistema traduce il formato interno dell'editor verso lo standard IMS/SCORM, crea il package e registra il nuovo corso fra i corsi disponibili.
Percorso alternativo	
Postcondizione	Il progetto è nello stato Saved.
Percorso d'eccezione	
Altri	

Use case	Edit project
Precondizione	- Essere nella pagina di gestione del progetto - Avere i diritti di scrittura sul progetto - Il progetto deve essere nello stato Opened - Il particolare utente deve aver in precedenza aperto il progetto.
Percorso base	1. L'utente entra nell'editor di progetto. 2. Il sistema avvia l'editor di progetto.
Percorso alternativo	
Postcondizione	L'utente è nell'editor di progetto
Percorso d'eccezione	
Altri	

- **Gestione risorse**: pagina che simula un comune file manager per poter organizzare i files delle proprie risorse come se si trattasse di un disco locale. I files vengono tuttavia immagazzinati nel sistema.



Use case	Cut Folder
Precondizione	Essere nel gestore delle risorse ed avere selezionato una folder
Percorso base	1.Premere il pulsante Cut Folder per tagliare la folder selezionata.
Percorso alternativo	
Postcondizione	
Percorso d'eccezione	Se l'utente non ha diritti per effettuare l'operazione viene visualizzato un messaggio d'errore.
Altri	Una volta che la folder è stata tagliata è possibile incollarla mediante Paste to Folder.

Use case	Paste to folder.
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una folder - Aver precedentemente tagliato una risorsa o una cartella - Avere i diritti di scrittura sulla cartella corrente.
Percorso base	1. Premdere il pulsante Paste to folder per incollare nella cartella corrente una carlte o una risorsa precedentemente tagliati.
Percorso alternativo	
Postcondizione	
Percorso d'eccezione	Se l'utente non ha diritti per effettuare l'operazione viene visualizzato un messaggio d'errore.
Altri	Per tagliare una risorsa Cut Resource, per tagliare un cartella Cut Folder.

Use case	Delete folder
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una folder - Avere i diritti di scrittura sulla cartella corrente
Percorso base	1. L'utente pereme il pulsante Dolete per cancellare la cartella corrente a tutti i contenuto in modo ricorsivo. 2. Il sistema visualizza un messaggio di avviso chiedendo conferma dell'operazione. 3. L'utente conferma.
Percorso alternativo	3.a L'utente non conferma ritornando alla situazione precedente l'operazione.
Postcondizione	
Percorso d'eccezione	Se l'utente non ha diritti per effettuare l'operazione viene visualizzato un messaggio d'errore.
Altri	

Use case	Select folder
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una folder - Avere i di lettura sulla cartella che si seleziona
Percorso base	1. L'utente seleziona una cartella dalla struttura ad albero delle cartelle delle risorse opppura dal contenuto di una cartella già selezionata. 2. Il sistema visualizza informazioni relative alla cartella e mostra il

	contenuto della cartella.
Percorso alternativo	
Postcondizione	
Percorso d'eccezione	Se l'utente non ha diritti per effettuare l'operazione le informazioni della cartella sono visualizzate ugualmente ma il contenuto della cartella non è visibile.
Altri	

Use case	View Folder Contents
Altri	Vedere Select Folder

Use case	View Edit Folder properties
Precondizione	- Avere selezionato una cartella mediante Select Folder.
Percorso base	1. L'utente preme il pulsante Edit corrispondente al valore che vuole modificare 2. Il sistema visualizza un textbox e i pulsanti Update e Cancel. 3. L'utente modifica il valore nella textbox . 4. L'utente preme Update. 5. Il sistema aggiorna vecchia valore con il valore inserito nella textbox.
Percorso alternativo	4.a L'utente preme Cancel. 5.a Il sistema ritorna allo stato precedente l'esecuzione dell'operazione.
Postcondizione	
Percorso d'eccezione	
Altri	

Use case	Upload resource to folder
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una folder. - Avere i diritti di scrittura sulla cartella corrente.
Percorso base	1. L'utente preme il pulsante Upload Resource 2. Il sistema visualizza una form per l'inserimento dei dati della risorsa di cui si vuole fare l'upload compreso il nome del file locale. 3. L'utente compila la form e preme il pulsante Submit 4. Il sistema esegue l'upload della risorse rendendola visibile nei contenuti della cartella corrente
Percorso alternativo	3.a L'utente preme Cancel 4.a Il sistema ritorna allo stato precedente l'esecuzione dell'operazione.
Postcondizione	
Percorso d'eccezione	Se l'utente non ha diritti per effettuare l'operazione viene visualizzato un messaggio d'errore.
Altri	

Use case	Upload streaming media to folder
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una folder. - Avere i diritti di scrittura sulla cartella corrente.
Percorso base	1. L'utente vuole eseguire l'upload di un media di grandi dimensioni che possa essere fruito in seguito in modalità di streaming. L'utente preme il pulsante Upload streaming media. 2. Il sistema visualizza una form per l'inserimento dei dati della risorsa di cui si vuole fare l'upload compreso il nome del file locale. 3. L'utente compila la form e preme il pulsante Submit 4. Il sistema esegue l'upload del meida rendendolo visibile nei contenuti della cartella corrente. Il sistema pubblica il file nello streaming server.
Percorso alternativo	3.a L'utente preme Cancel. 4.a Il sistema ritorna allo stato precedente l'esecuzione dell'operazione.
Postcondizione	
Percorso d'eccezione	Se l'utente non ha diritti per effettuare l'operazione viene visualizzato un messaggio d'errore.
Altri	

Use case	Gestione ACL folder
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una folder. - Avere i diritti di scrittura sulla cartella corrente o essere membro del gruppo Administrator.
Percorso base	Vedere Use case generale per gestione ACL.
Percorso alternativo	
Postcondizione	
Percorso d'eccezione	Se l'utente non ha diritti per effettuare l'operazione viene visualizzato un messaggio d'errore.
Altri	

Use case	Select resource
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una folder. - Avere i diritti di lettura sulla cartella corrente.
Percorso base	1. L'utente seleziona una risorsa dalla lista di elementi (risorde e cartelle) della cartella corrente. 2. Il sistema visualizza la pagina della risorsa selezionata.
Percorso alternativo	
Postcondizione	
Percorso d'eccezione	
Altri	

Use case	Delete resource
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una risorsa. - Avere i diritti di scittura sulla cartella che contiente la risorsa selezionata.

Percorso base	1. L'utente preme il pulsante Delete. 2. Il sistema chiede conferma dell'operazione tramite messaggio. 3. L'utente preme Ok 4 Il sistema cancella la risorsa e visualizza la pagina della cartella che conteneva la risorsa appena cancellata.
Percorso alternativo	3.a L'utente preme No 4.a Il sistema ritorna allo stato precedente l'operazione.
Postcondizione	
Percorso d'eccezione	
Altri	

Use case	Preview resource
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una risorsa.
Percorso base	1. L'utente vuole una preview della risorsa. Preme il pulsate Preview 2. Il sistema esegue il dowload della risorsa. Se il file viene aperto salvato su disco locale dell'utente è deciso dall'utente o da una impostazione del browser.
Percorso alternativo	
Postcondizione	
Percorso d'eccezione	
Altri	

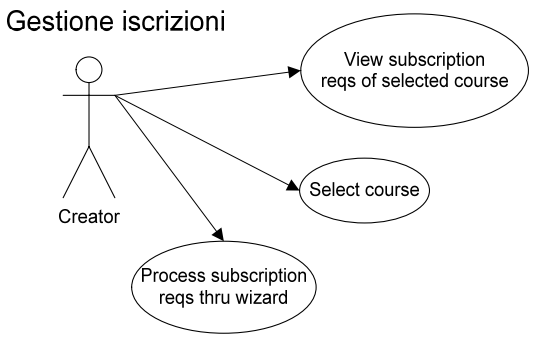
Use case	Dowload resource.
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una risorsa. - Avere i diritti di scittura sulla cartella che contiente la risorsa selezionata.
Percorso base	1. L'utente vuole scaricare una copia del file della risorsa dal sistema. Preme il pulsate Download. 2. Il sistema esegue il dowload della risorsa. Se il file viene aperto salvato su disco locale dell'utente è deciso dall'utente o da una impostazione del browser.
Percorso alternativo	
Postcondizione	
Percorso d'eccezione	
Altri	

Use case	Cut resource
Precondizione	- Essere nel gestore delle risorse ed avere selezionato una risorsa. - Avere i diritti di scrittura sulla cartella contenente la risorsa.
Percorso base	1. L'utente vuole spostare una risorsa da una cartella all'altra. Preme il pulsante Cut resource.
Percorso alternativo	
Postcondizione	

Percorso d'eccezione	Se l'utente non ha diritti per effettuare l'operazione viene visualizzato un messaggio d'errore.
Altri	Una volta che la risorsa è stata tagliata è possibile incollarla mediante Paste to Folder.

Use case	View / Edit resource properties
Precondizione	Avere selezionato una risorsa.
Percorso base	1. L'utente preme il pulsante Edit corrispondente al valore che vuole modificare 2. Il sistema visualizza un textbox e i pulsanti Update e Cancel. 3. L'utente modifica il valore nella textbox . 4. L'utente preme Update. 5. Il sistema aggiorna vecchia valore con il valore inserito nella textbox.
Percorso alternativo	4.a L'utente preme Cancel. 5.a Il sistema ritorna allo stato precedente l'esecuzione dell'operazione.
Postcondizione	
Percorso d'eccezione	

- **Gestione iscrizioni**: pagina che propone principalmente un wizard per guidare l'utente nella distribuzione dei diritti (Vedi sezione 6.3). Il concetto di iscrizione è astratto, in quanto non esiste nel sistema una vera e propria iscrizione ad un corso, e dinamico, poiché viene determinata in base ai diritti che un utente ha effettivamente sul corso. Se un utente ha almeno i diritti di lettura e/o esecuzione allora risulta iscritto al corso. Per maggiori chiarimenti sul modello riguardante i diritti vedi sezione 6.2 e 6.3



Use case	Select course
Precondizione	Essere nella pagina di gestione delle richieste di sottoscrizioni.
Percorso base	1. L'utente seleziona un corso.
Percorso alternativo	
Postcondizione	L'utente è nella pagina di gestione delle richieste di sottoscrizioni.
Percorso d'eccezione	
Altri	

Use case	View subscription requests of selected course
Precondizione	Essere nella pagina di gestione delle richieste di sottoscrizioni ad un corso.
Percorso base	1. L'utente seleziona un corso 2. Il sistema avvisa l'utente che per il corso selezionato ci sono delle richieste di sottoscrizioni pendenti
Percorso alternativo	1. L'utente può cancellare le richieste di sottoscrizione.
Postcondizione	L'utente è nella pagina di gestione delle richieste di sottoscrizioni ad un corso.
Percorso d'eccezione	
Altri	

Use case	Process subscription requests.
Precondizione	Avere selezionato subscription requests.
Percorso base	1. L'utente decide di gestire le richieste di sottoscrizione. Tali richieste possono essere filtrate per proprietà del corso e dell'utente (Ad esempio "solo richieste per miei utenti" e/o "solo rischiate per i miei corsi"). 2. Il sistema avvia il wizard che guida l'utente nelle varie fasi per l'esecuzione delle operazioni necessarie.
Percorso alternativo	
Postcondizione	L'utente è nella pagina di gestione delle richieste di sottoscrizioni.
Percorso d'eccezione	
Altri	Vedere Figura 25 per dettagli relativi al wizard.

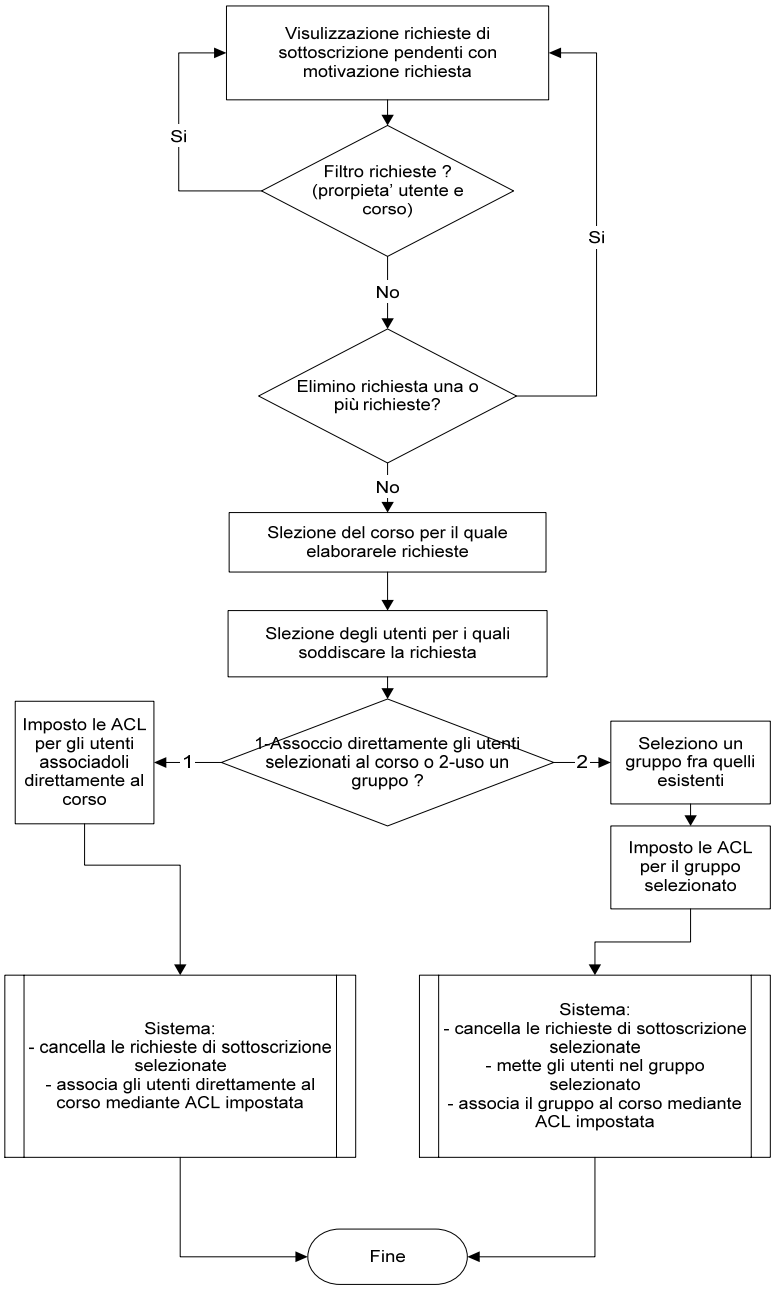


Figura 25: diagramma che descrive i passi e le azioni che possono avvenire durante la fase di gestione delle richieste di sottoscrizione fatte dagli utenti del sistema. Tale processo guidato ha il compito di semplificare la fase di distribuzione dei diritti agli utenti tramite ACL . Senza una procedura guidata e automatizzata la distribuzione dei diritti, soprattutto per un grande numero di utenti, risulterebbe macchinosa e ripetitiva.

6.4 Classi e strutture dati

I casi d'uso della sezione precedente suggeriscono la creazione delle seguenti classi per modellare le entità coinvolte.

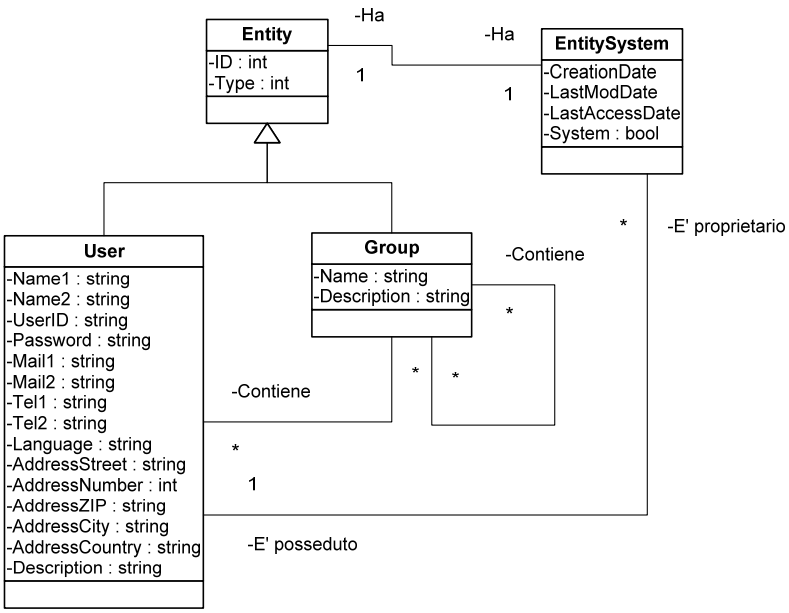


Figura 26: Diagramma concettuale delle classi relative a utenti e gruppi

La classe Entity è padre delle classi User e Group. Questa scelta è fatta per avere un unico identificatore per le diverse istanze di User e Group in modo da poter gestire facilmente l'annidamento di gruppi. Esisterà infatti un'unica tabella nel database avente due colonne entrambe contenenti un Entity.ID in modo da poter porre in un gruppo uno o più gruppi. Si è scelto inoltre di separare dall'oggetto Entity le informazioni relative al sistema creando una classe con rapporto uno a uno e nome EntitySystem. In tale classe è specificato il proprietario dell'istanza di Entity che sarà un'istanza di User. Vedi Figura 26.

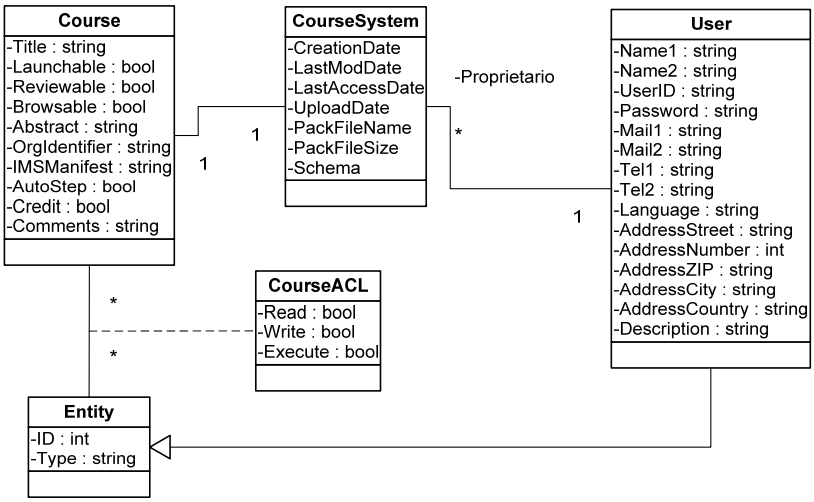


Figura 27: Diagramma concettuale delle classi relative alla relazione fra utenti, gruppi e corsi.

Anche per la classe Course le informazioni relative al sistema sono state separate in una classe separata CourseSystem avente rapporto uno a uno. Ogni istanza della classe Course ha un singolo proprietario istanza della classe User. La relazione utilizzata per la distribuzione dei diritti sul corso utilizza la classe Entity come estremo in quanto ad un corso è possibile associare oltre che uno o più utente anche uno o più gruppi in modo che tutti gli utenti appartenenti ad un particolare gruppo ereditino i diritti specificati nella CourseACL implicata nella relazione.
Per rendere ancora più flessibile e scalabile il sistema è possibile mappare il servizio di distribuzione dell'architettura generica dell'LMS di Figura 1 su più server fisici di distribuzione dei contenuti. Una esigenza implementativa di questo tipo condiziona la fase di progettazione. Tenendo conto di questo aspetto il diagramma di Figura 27 può essere esteso portando al diagramma di Figura 28 introducendo la classe Server e le sue relazioni con le altre entità .

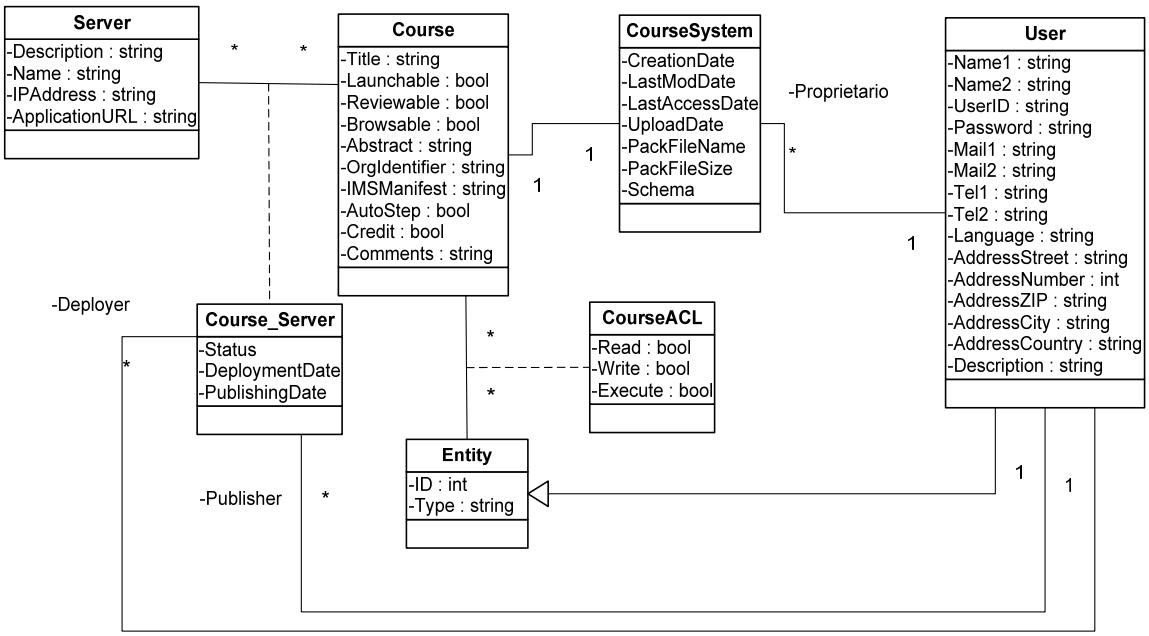


Figura 28:diagramma delle classi tenendo conto della possibilità di avere più delivery service contemporaneamente.

Per supportare i corsi SCORM e tenere traccia delle interazioni dell'utente con gli SCO che li costituiscono è necessario introdurre una classe che associa uno SCO di un corso con un utente. Tale struttura dati mantiene lo stato dello SCO durante l'esecuzione memorizzando la struttura CMI che viene letta e scritta dal run-time durante l'esecuzione dello SCO.

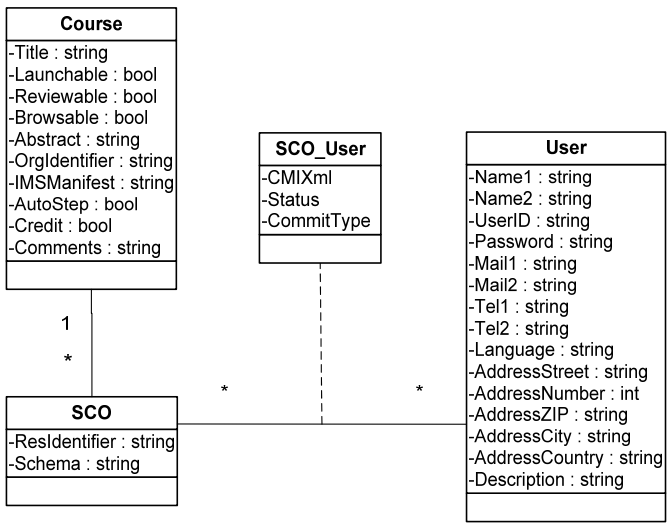


Figura 29: diagramma delle classi con la relazione fra utente e SCO per il mantenimento dello stato di esecuzione degli SCO dei corsi SCORM.

6.4.1 Strutture dati per i modelli[33]

Le struttura dati descritte nelle specifiche IMS SCORM sono facilmente rappresentabili utilizzando l'XML come modello. Infatti gli stessi team che hanno prodotto IMS e SCORM utilizzano efficacemente tale tecnologia sia nelle specifiche formali che nei dettagli di tipo implementativo. E' naturale, quindi che anche lezi.NET utilizzi XML nella maggior parte dei modelli e come formato di interscambio di dati.

Per facilitare la successiva fase di sviluppo ed implementazione si è pensato di creare delle interfacce orientate agli oggetti per ogni documento XML utilizzato durante l'esecuzione dell'applicazione. Lo scopo di queste API è quello di permettere di avere una interfaccia semplice che nasconda i dettagli implementativi relativi all'utilizzo di API per l'accesso a documenti XML fornite dal framework .NET .

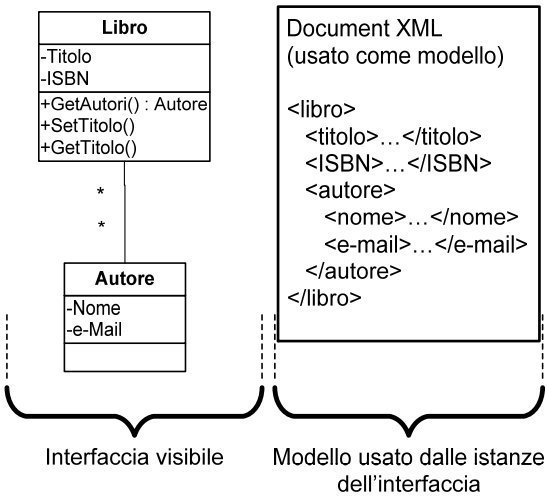


Figura 30: Esempio su come fornire una interfaccia OO ad un documento XML. L'interfaccia costituisce un wrapper (API) al documento nascondo i dettagli del modello.

Struttura CMI. Struttura dati utilizzata dal run-time per mantenere lo stato dell'esecuzione di uno SCO da parte di un utente. La struttura dati alla quale si fornisce un'interfaccia OO è il modello CMI di cui si è parlato nella Sezione 3.2.5. Anche in questo caso, un modo naturale di rappresentare la struttura dati che implementa CMI è un documento XML. Ad esempio, per memorizzare il dato dell'elemento "cmi.core.student_name"può essere utilizzato il seguente documento XML.

```
<xml version="1.0">
            <item>
                  <name>cmi</name>
                  <item>
                        <name>core</name>
                        <element>
                              <name>student_name</name>
                              <value>Stefano Bruna</value>
                        </element>
                  </item>
            </item>
</xml>
```

[33] *Il termine modello è utilizzato come sinonimo di struttura dati. Tale termine viene utilizzato poiché, spesso, l'architettura di leziNET è basata sul famoso design pattern Model-View-Controller nel quale il modello viene visto come il modulo all'interno del quale è contenuto lo stato, e quindi la struttura dati.*

Questo documento non contiene informazioni sul tipo di dati, il tipo di accesso e tutte le altre informazioni che caratterizzano i dati della struttura CMI . Tali informazioni sono implicitamente contenute nelle classi dell'API che si preoccupano di verificare tutti i vincoli imposti dalle specifiche di CMI:

- Controllo del tipo dei dati che vengono letti e scritti nelle varie fogli della struttura.
- Controllo del tipo di accesso che viene eseguito durante le operazioni di lettura e scrittura. (lettura-scrittura / solo lettura / solo scrittura).
- Controllo della cardinalitá dei rami della struttura.
- Aggiornamento dei campi che contengono contatori per i rami aventi cardinalitá variabile. Ad esempio, in CMI esiste il seguente ramo: cmi.iteractions.n.iteraction dove n può andare da 0 a infinito. L'API per CMI deve aggiornare il campo cmi.iteractions._count con il numero di interaction che sono effettivamente usate.
- Controllo dei passaggi di stato di alcuni elementi con particolari vincoli sui valori. Ad esempio, il valore di "cmi.core.lesson_status" può passare solamente da "Not attempted", a "Browsed" ma non viceversa.
- Aggiornamento di elementi derivati funzione di altri elementi

Si possono quindi intuire tutti i vantaggi di una soluzione di questo tipo. Se non utilizzata, per ogni accesso al modello CMI sarebbe stato necessario, da parte dello sviluppatore, scrivere codice per effettuare i controlli descritti sopra. Invece utilizzando un API che "circonda" il modello per accedere, ad esempio allo stato della lezione basta il seguente codice:

```
CMICmi cmi = new CMICmi(…)
cmi.LoadXML(…) // caricamento del modello
CMIElement element = cmi.GetValue("cmi.core.student_name");
String sname = element.Value;
```

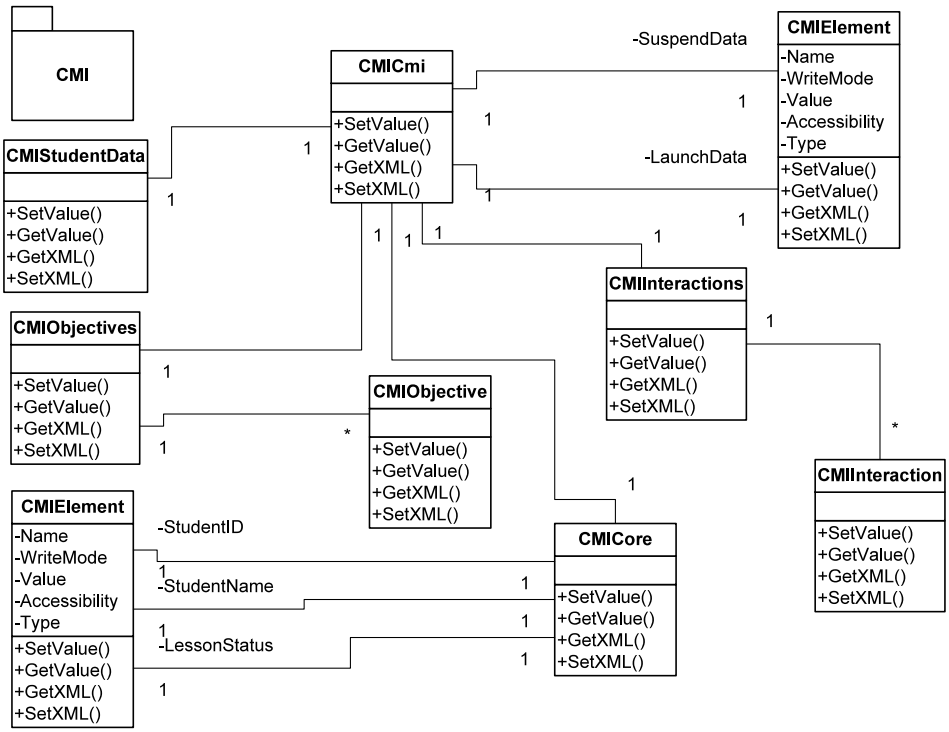


Figura 31: questo diagramma delle classi é parziale. Il diagramma completo é composto da 39 classi aventi principalmente realzioni1:1 o 1:. Tutti gli elementi foglia della struttura ad albero CMI sono rappresentati dalla classe CMIElement che viene inizializzata diversamente dal padre a seconda del tipo di dato che dovrà contenere .*

Esempio di possibile implementazione:
Lo SCO in esecuzione, tramite il run-time vuole leggere il nome dell'utente per poterlo scrivere in uno SCO di benvenuto introduttivo del corso. Per fare questo lo SCO chiama il metodo del run-time LMSGetValue("cmi.core.student_name"). Tale richiesta viene gestita dall'LMS che chiama il metodo GetValue("cmi.core.student_name") su una istanza della classe CMICmi. Quato metodo esegue il parsing della stringa in dot-notation eliminando la prima parte prima del primo punto (cmi) e chiama lo stesso metodo dell'istanza della classe CMICore.Quest'ultima , vedendo che "student_name" è un proprio elemento figlio chiede il valore al CMIElement. Il ritorno da queste chiamate di metodi annidati permette di ottenere il valore richiesto.

Struttura IMSManifest. E' stato necessario sviluppare naturalmente un' API, funzionante in sola lettura, per poter avere un'interfaccia OO al documento imsmanifest.xml descritto dettagliatamente nella sezione 3.1.2. La creazione di tale API è giustificata dal fatto che spesso le pagine devono avere un accesso veloce ed efficiente alle informazioni contenute nel manifest di un corso per poter visualizzare, ad esempio, la tabella dei contenuti del corso. Per soddisfare i requisiti prestazionali e per il fatto che il manifest di un corso è un documento che può solo essere letto è conveniente duplicare il modello: uno residente nel file del documento xml e l'altro in memoria come membri delle istanze di classi fornite dall'API. Grazie ad una soluzione di questo tipo, durante l'elaborazione di una pagina aspx, il parsing di un particolare campo del documento XML sarà eseguito solo la prima

volta. Per tutti gli accessi seguenti verranno utilizzate le informazioni, in un certo senso "cachate"[34] , nelle strutture dati delle istanze delle classi fornite dall'API. Vedi Figura 32.

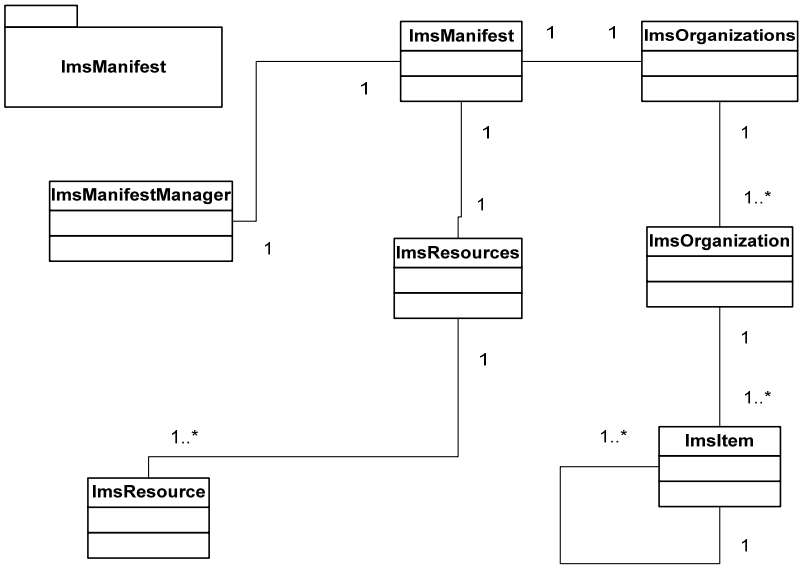


Figura 32: Diagramma delle classi delle API per la gestione (operazioni di sola lettura) del documento XML imsManifest.xml dello standard IMS.

Struttura leziManifest. Per supportare la creazione di progetti e poi il build del progetto per la produzione di un corso è necessario un modello che permetta la memorizzazione della struttura del progetto. Utilizzando le informazioni di tale modello un traduttore crea il manifest.xml e i vari file html del corso utilizzando, eventualmente, dei template. Le informazioni necessarie per l'esecuzione dell'editor sono necessariamente maggiori delle informazioni contenute nel prodotto finale cioè il manifest del corso. Dal punto di vista del package e dello standard IMS una fotografia non é altro che un file di una risorsa. Invece per l'editor potrebbe trattarsi, ad esempio, di una slide associata ad un particolare filmato che deve essere visualizzata quando viene lanciato un particolare evento dal filmato etc. E' quindi possibile scegliere due strade differenti: una è quella di estendere il manifest definito da IMS integrato mediante nuovi tags ed eventualmente un nuovo namespaces per poter memorizzare i dati necessari al funzionamento dell'editor, oppure creare un documento xml separato che viene utilizzato in fase di editing ed utilizzato successivamente per la produzione del manifest ufficiale.

[34] *Dall'inglese to cache. Seppur sgradevoli per l'amante dell'italiano, queste parole inglesi brutalmente italianizzate permettono di emulare la buona capacità sintetica dell'inglese quando si scrive o parla in italiano, soprattutto in ambito tecnico.*

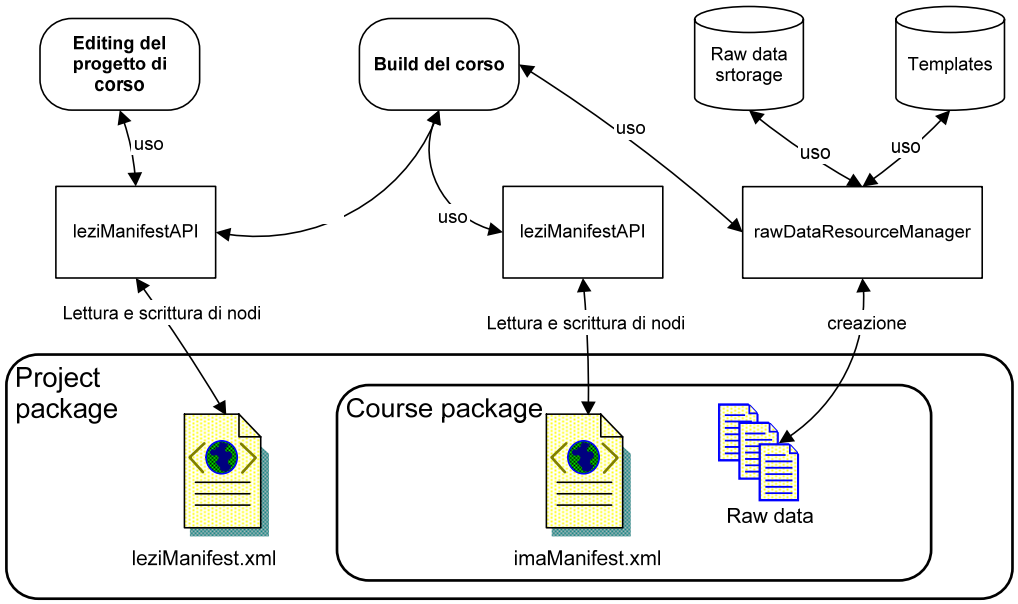


Figura 33: Schema che riassume le risorse che vengono utilizzate durante la fase di build del corso partendo dal leziManifest prodotto durante la fase di editing.

La seconda soluzione pur non sfruttando una ottima caratteristica dell'xml quale l'estendibilitá di un documento, sembra essere una soluzione più pulita in quanto non appesantisce il manifest ed evita di dover produrre una unica complessa API in grado di modificare il documento anche durante la fase di editing. Le operazioni che coinvolgono i due tipi di documenti possono essere riassunte in Figura 33. E' possibile notare come l'output dell'operazione di build del corso produca un project package contenente il package vero e proprio del corso. Tuttavia, il project package, pur contenendo anche le informazioni per il corretto funzionamento dell'editor, è un package che risponde alle specifiche IMS e quindi questo stesso package potrà essere considerato il package del corso prodotto.
In figura 34 viene rappresentato il digramma delle classi dell'api per il leziManifest mentre nell'Esempio 14 è possibile vedere un esempio di documento leziManifest.xml.

Esempio 14. L'insieme di attributi appartenenti all'elemento radice del documento utilizzati per la produzione di un identificatore univoco per i vari elementi creati durante l'editing del corso. Ad esempio, l'attributo rpn (resource progress number) è arrivato al valore 2, infatti l'ultima resource creata ha l'identificatore "resource_1', quindi la prossima risorsa aggiunta dall'editor avrà identificatore "resource_2" e l'attributo rpn verrà incrementato di uno.

```
<lezimanifest ipn="3" spn="8" rpn="2" dpn="1" lpn="1">
  <items>
    <item identifier="item_0">
      <title>Lezione 1</title>
      <item identifier="item_1" resourceidentifierref="resource_0">
        <title>Introduzione</title>
      </item>
      <item identifier="item_2" resourceidentifierref="resource_1">
        <title>Contenuti</title>
      </item>
    </item>
  </items>
  <resources>
    <resource identifier="resource_0" templatetype="mmt" type="webcontent">
      <video href="resource_0/Media.asx" streaming="True">
```

```
        <title>video3</title>
        <description>
        </description>
      </video>
      <slides>
        <slide identifier="slide_0" number="1" href="resource_0/slide_0.jpg"
filename="slide_0.jpg">
          <name>Crop 1</name>
          <description>
          </description>
        </slide>
        <slide identifier="slide_1" number="2" href="resource_0/slide_1.jpg"
filename="slide_1.jpg">
          <name>Crop 2</name>
          <description>
          </description>
        </slide>
        <slide identifier="slide_2" number="3" href="resource_0/slide_2.jpg"
filename="slide_2.jpg">
          <name>Crop 3</name>
          <description>
          </description>
        </slide>
        <slide identifier="slide_3" number="4" href="resource_0/slide_3.jpg"
filename="slide_3.jpg">
          <name>Crop 4</name>
          <description>
          </description>
        </slide>
      </slides>
      <links>
        <link identifier="link_0">
          <name>Intel</name>
          <description>
          </description>
          <url>http://www.intel.com</url>
        </link>
      </links>
      <docs>
        <doc identifier="doc_0" href="resource_0/doc_0.pdf" filename="doc_0.pdf">
          <name>satellitepro_6100</name>
          <description>
          </description>
        </doc>
      </docs>
    </resource>
    <resource identifier="resource_1" templatetype="mmt" type="webcontent">
      <video href="resource_1/Media.asx" streaming="True">
        <title>video3</title>
        <description>
        </description>
      </video>
      <slides>
        <slide identifier="slide_4" number="1" href="resource_1/slide_4.jpg"
filename="slide_4.jpg">
          <name>Crop 1</name>
          <description>
          </description>
        </slide>
        <slide identifier="slide_5" number="2" href="resource_1/slide_5.jpg"
filename="slide_5.jpg">
          <name>Crop 2</name>
          <description>
          </description>
        </slide>
        <slide identifier="slide_6" number="3" href="resource_1/slide_6.jpg"
filename="slide_6.jpg">
          <name>Crop 3</name>
          <description>
```

```
        </description>
      </slide>
      <slide identifier="slide_7" number="4" href="resource_1/slide_7.jpg"
filename="slide_7.jpg">
        <name>Crop 4</name>
        <description>
        </description>
      </slide>
    </slides>
  </resource>
</resources>
</lezimanifest>
```

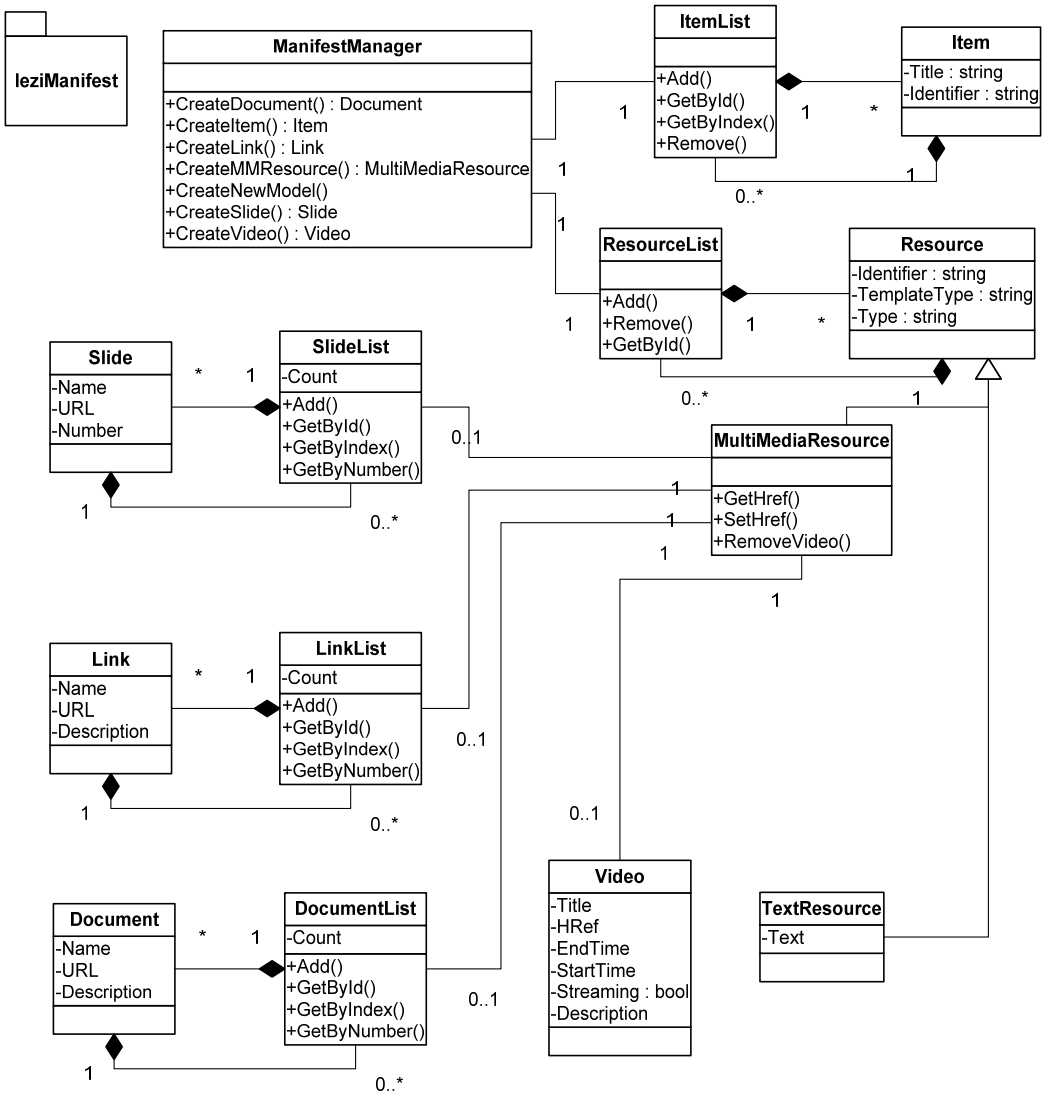


Figura 34: Diagramma delle classi messe a disposizione dall'interfaccia.

La maggior parte delle API e delle classi precedentemente descritte devono essere utilizzabili nelle diverse parti dell'applicazione. Per questo motivo è utile includerle in una libreria, una dll .NET. che sarà poi utilizzata dalle applicazioni che costituiscono l'intero sistema.

7 Architettura

7.1 Struttura funzionale generale

L'architettura funzionale dell'applicazione utilizza come riferimento il modello di LMS proposto da SCORM e visto nella sezione 3.2.1. Lezi.NET implementa un versione semplificata di tale modello conservando le funzionalità essenziali, soprattutto quello legate all'ambiente di esecuzione. Inoltre i requisiti impongono di estendere il modello di LMS introducendo due funzionalità: la gestione delle risorse (files e documenti usati dagli autori per creare corsi) e la gestione dei progetti cioè l'editing dei corsi. Queste due funzionalità non vengono probabilmente incluse nel modello di LMS tradizionale poiché la fase di authoring dei package dei corsi è considerata come separata e può avvenire "off-line". Tuttavia si ricorda che il processo che lezi.NET tenta di realizzare è in breve il seguente:

1 - Raccolta di contenuti e loro organizzazione
2 - Creazione di un corso, utilizzando i contenuti raccolti
3 - Deployment del corso e verifica
4 - Pubblicazione del corso
5 - Fruizione del corso

Il poter eseguire tutte queste fasi utilizzando la medesima applicazione Web-based ha i seguenti vantaggi (oltre a quelli che si hanno con la fruizione di contenuti on-line) :

- Unico ambiente di lavoro per il "creatore" e per il "fruitore" di corsi. Questo comporta il dover imparare ad avere a che fare con una sola applicazione e un solo tipo di interfaccia.
- Unica login ad un sistema sicuro che protegge i contenuti che raccoglie. I documenti di un creatore sono probabilmente memorizzati su un server cha ha meccanismi per garantire sia la disponibilità che la sicurezza dei dati.
- L'avere a disposizione un editor di corsi Web-based permette di evitare l'installazione di applicazioni presso le workstations degli autori riducendo così notevolmente i costi di deployment e di manutenzione.

Tutto l'LMS comprendente di gestore delle risorse e editor di corsi sono accessibili dovunque si sia una connessione al Web.

Tenendo conto di questi diversi aspetti è possibile elaborare l'architettura funzionale di Figura 35.

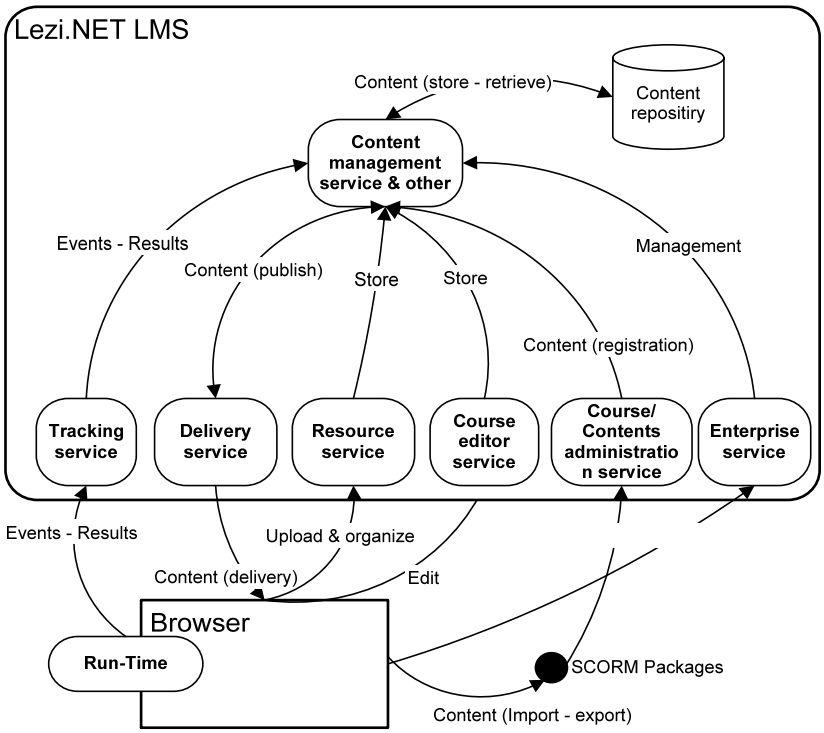


Figura 35 : Architettura funzionale di lezi.NET

I diversi componenti funzionali di Figura X sono i seguenti:

- **Tracking service**: servizio chiamato dal run-time per permettere l'interazione fra contenuto e LMS. Tutte le informazioni riguardanti la particolare fruizione di uno SCO da parte di un utente vengono raccolte da questo servizio che le trasmette, per una eventuale memorizzazione, al Content management service (CMS).
- **Delivery service**: servizio che si occupa della distribuzione dei contenuti. I contenuti presenti nel CMS vengono precedentemente pubblicati presso questo servizio.
- **Resource service**: servizio che gestisce l'upload, l'organizzazione e la memorizzazione delle risorse utilizziate dall'editor per la creazione di corsi. La memorizzazione, in particolare sarà un servizio richiesto a sua volta al CMS. Permette la distribuzione di diritti a utenti e gruppi per le risorse tramite ACL.
- **Course editor service**: servizio per la creazione di corsi, richiede le risorse, precedentemente gestite dal Resource service, al CMS e le utilizza per la produzione di corsi (package IMS/SCORM) che verranno memorizzati dal CMS. Permette la distribuzione di diritti agli utenti e gruppi per i progetti tramite ACL.
- **Course contents administration service**: servizio per gestire il deployment, la pubblicazione e cancellazione di corsi. Permette la distribuzione di diritti a gruppi ed utenti per i corsi tramite ACL.
- **Enterprise service**: gestisce gruppi ed utenti del sistema. Permette di gestire il profilo personalizzato degli utenti. Fornisce un meccanismo per gestire le richieste di sottoscrizione ai corsi.
- **Content management service**: gestisce l'intero stato del sistema, fornisce in pratica ai servizi di più alto livello, funzionalità per lo storage di tutte le informazioni. Ad esempio il CMS pilotato dal Course / Content administration service può pubblicare sul delivery service un corso.

- **Browser e run-time**: fornisce l'ambiente client-side all'utente per la fruizione e il tracciamento degli eventi e dei dati derivanti dall'interazione con i contenuti.

7.2 Struttura fisica generale

Nella precedente architettura funzionale è possibile distinguere fra funzionalità e servizi di alto livello, cioè quelli che, predisponendo un'interfaccia utente, forniscono servizi direttamente all'utente e servizi, il CMS, che forniscono un supporto ai precedenti. Questa suddivisione funzionale influisce naturalmente sulla struttura fisica dell'applicazione portando all'architettura di Figura 36.

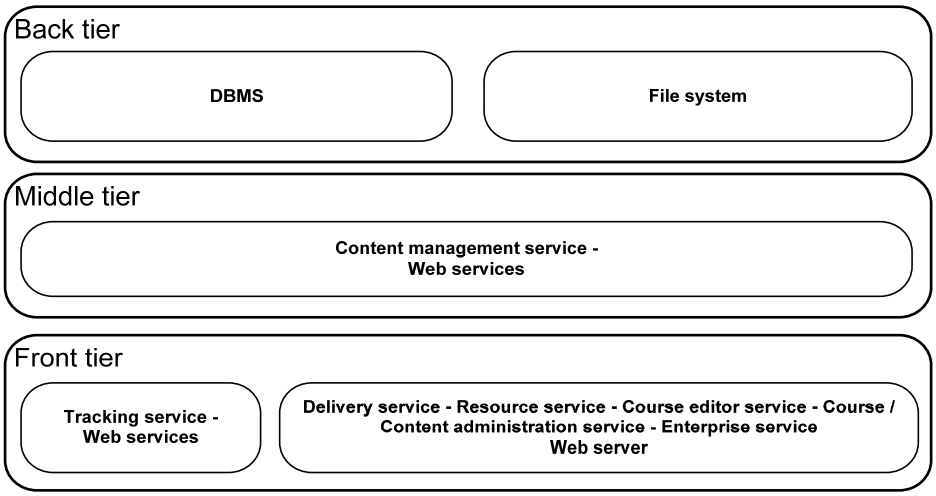


Figura 36: architettura fisico - funzionale. Le diverse unita di servizio sono state allocate in posizioni fisiche in una tipica struttura multi tier.

- Back tier:

Il back tier ospita un DBMS ed eventualmente servizi per la memorizzazione di dati su file system. E' stato scelto questo tipo di memorizzazione per effettuare lo storage dei package dei corsi e delle risorse degli utenti utilizzando poi un riferimento contenuto del database per indicizzarli. Sarebbe stato possibile anche memorizzare tali files direttamente nel DBMS mediante una tecnologia ormai supportata dalla maggior parte di DBMS commerciali che permette di inserire nei record delle tabelle file di grandi dimensioni. Ad esempio, Microsoft SQL Server 2000 permette di inserire files con dimensioni fino a 3 GB. I files memorizzati in questo modo sono detti BLOB. Entrambe le soluzioni hanno vantaggi e svantaggi. La memorizzazione su file system con indicizzazione su database ha il vantaggio di non appesantire il database e di sfruttare la maggiore velocità di accesso e memorizzazione del file system senza restrizioni di dimensione alcuna. Un eventuale cambio di posizione dei files comporta, purtroppo, un aggiornamento di tutti i riferimenti ai files nel database. Invece la memorizzazione diretta dei file nel database ha il vantaggio che spostando il database si sposta tutto. Tuttavia questa soluzione causa un incremento notevole delle dimensioni del database rallentandone sensibilmente le performance.
In lezi.NET ho adottato una soluzione intermedia. I files delle risorse e i package, come gia detto, sono memorizzati su file system. Nel database è presente un riferimento a tali file ed eventualmente alcuni metadati. Invece i diversi documenti XML che sono utilizzati

dall'applicazione sono "blobbati" direttamente nei record insieme agli altri dati. Questi documenti hanno comunque dimensioni di solito inferiori ai 100 KBytes.

- Middle tier

Il middle tier offre servizi al front tier nascondendo la complessità relativa alla memorizzazione e gestione delle risorse, dei corsi, dei progetti e di tutti gli altri oggetti del sistema. I servizi offrono le funzionalità di basso livello che riguardano:

- Creazione, cancellazione, modifica di utenti e gruppi.
- Creazione, cancellazione, modifica di proprietà, pubblicazione, deployment, download di corsi.
- Creazione, cancellazione, apertura, chiusura, editing, modifica di proprietà di progetti di corsi.
- Pubblicazione e gestione news e commenti.
- Trasferimento, cancellazione, modifica di proprietà di files di risorse archiviati.
- Archiviazione dei dati provenienti dal tracking service riguardanti l'interazione con l'utente durante la fruizione dei corsi.
- Memorizzazione e gestione delle Access control list delle diverse entità di sistema per distribuzione dei diritti.
- Autenticazione degli utenti e autorizzazione all'esecuzione delle diverse operazioni.

Il middle tier fornisce anche un ulteriore grado di sicurezza in quanto, per tutti i servizi che implementa, verifica sia autenticazione che autorizzazione. I servizi offerti dal CMS sono resi disponibili al front tier mediante Web services. Questa scelta implementativa è dovuta principalmente al desiderio di sperimentare una tecnologia emergente. Tale scelta porta a vantaggi come, ad esempio, la possibilità di interoperare facilmente con sistemi diversi, ma anche svantaggi soprattutto relativi alle performance. Una chiamata ad un Web services comporta, infatti, la creazione di messaggi SOAP-XML che sono meno efficienti dal punto di vista del payload. Infatti, a parità di dati da trasmettere la serializzazione binaria permette di trasmettere meno dati effettivi. Per questo motivo alcuni servizi del middle tier, come ad esempio il trasferimento dei package zip dei corsi, sono stati implementati usando sempre http come supporto, ma invece di usare i Web service, è stata utilizzata una normale codifica binaria uguale a quella che s'impiega quando si fa, ad esempio, il download o l'upload di files dal mediante il browser.

- Front tier

Il front tier fornisce servizi che possono essere raggruppati in due insiemi. Il tracking service, implementato tramite Web services fornisce la parte server side del run-time environment SCORM e l'insieme degli altri servizi, implementati mediante pagine attive .NET aspx, che forniscono la parte di "presentazione" dell'applicazione. Grazie alla centralizzazione dei servizi offerti dal Content management system del middle tier è possibile replicare facilmente il front tier. Una replica di questo tipo rende il l'intero sistema altamente scalabile. E' possibile quindi affiancare n front tier (Vedi Figura 37) aumentando così la "potenza" del sistema fino al momento in cui il CMS diventerà il collo di bottiglia . Fortunatamente questo accadrà con n abbastanza grande in quanto il CM offre servizi solo fra una sessione di fruizione e l'altra, ad esempio login dell'utente oppure visualizzazione della lista dei corsi, mentre per la maggior parte del tempo, durante la fruizione dei corsi, tutto il carico computazionale spetta la front tier

(replicabile), in particolare al tracking service. Nella sezione 9 saranno analizzate con maggiore dettaglio le performance dell'applicazione.

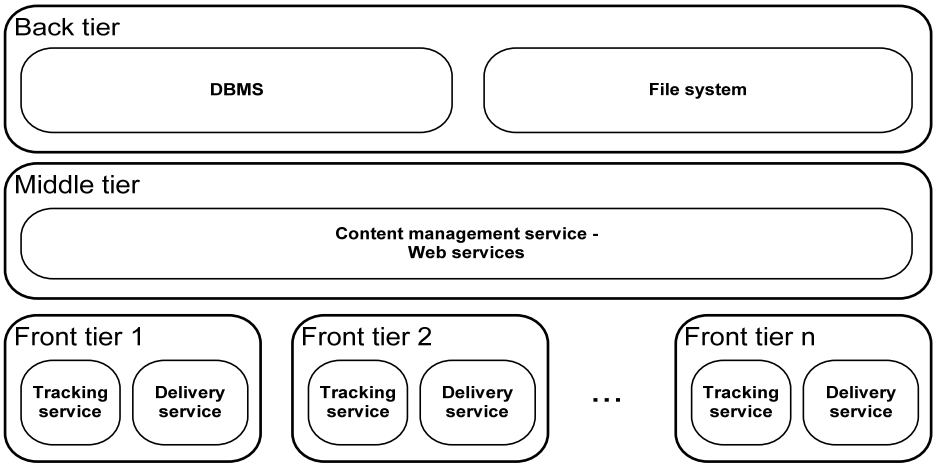


Figura 37: scalabilitá di lezi.NET: il front tier viene replicato aumentando disponibilità e numeri di client contemporanei.

La scelta di utilizzare fra il middle tier e il front tier sempre il protocollo http sulla porta 80 permette di avere una maggiore sicurezza di sistema. Infatti, è possibile porre un firewall fra i due tier in modo da isolare ulteriormente il CMS. Naturalmente anche i front tier possono essere protetti da firewall poiché tutti i canali di comunicazioni, sia quello del tracking service (Web service) che quello utilizzato da tutti gli altri servizi (http + html) utilizzano un solo protocollo e una sola porta. Eventualmente in caso di streaming sarà necessario aprire altre porte sul firewall. In Figura 38 è possibile vedere l'architettura fisica di leziNET in cui vengono specificate anche le scelte che riguardano la particolare piattaforma software.

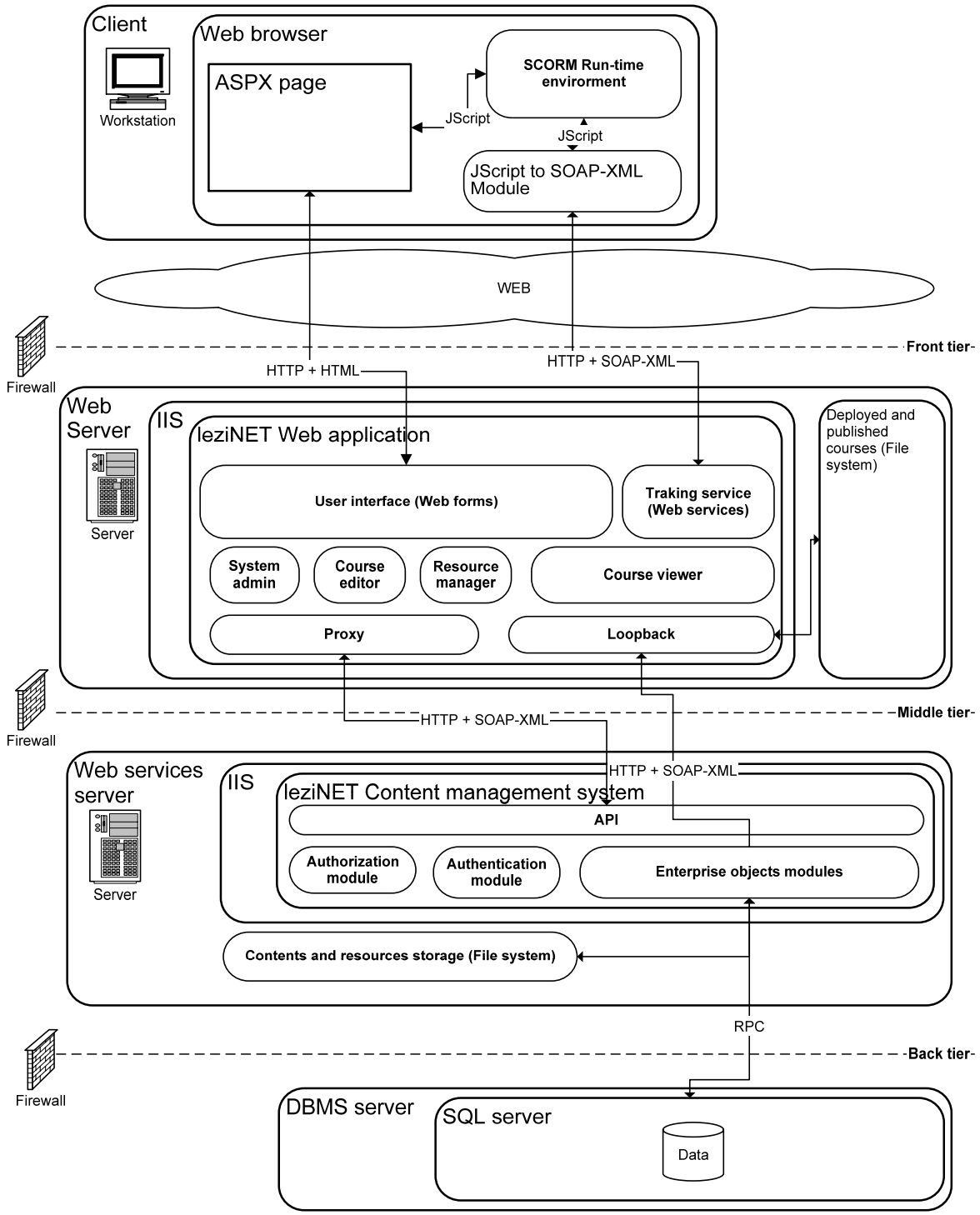


Figura 38: architettura con indicazioni di deployment dell'applicazione.

7.3 Struttura della pagina

La piattaforma .NET permette, come visto nella sezione 4.2, di utilizzare i concetti e le caratteristiche della programmazione orientata agli oggetti. La separazione in due files della presentazione (file: nome_della_pagina.aspx) e della logica (file: nome_della_pagina.cs) permette organizzare un'architettura base comune a tutte le pagine Web dell'applicazione. Da punto di vista logico ogni pagina può essere vista come un insieme di componenti che cooperano per fornire all'utente un'interfaccia che gli permetta di interagire con il sistema.

Questi componenti, che possono essere fra loro molto diversi, devono però essere supportati da un contenitore che fornisca una sorta di ambiente in cui possano essere eseguiti. Questo ambiente deve fornire i seguenti servizi:

- Sistema ad eventi per la visualizzazione di messaggi di errore: ogni componente può incontrare situazioni di errore dovute a difetti o particolari stati del sistema oppure ad una non corretta interazione dell'utente. In situazioni di questo tipo, il componente chiama il servizio di gestione degli errori della pagina che lo contiene. Sarà la pagina ad occuparsi della visualizzazione del messaggio. In questo modo oltre a fornire un'interfaccia comune, non è necessario implementare la gestione di situazioni di questo tipo per ogni componente.

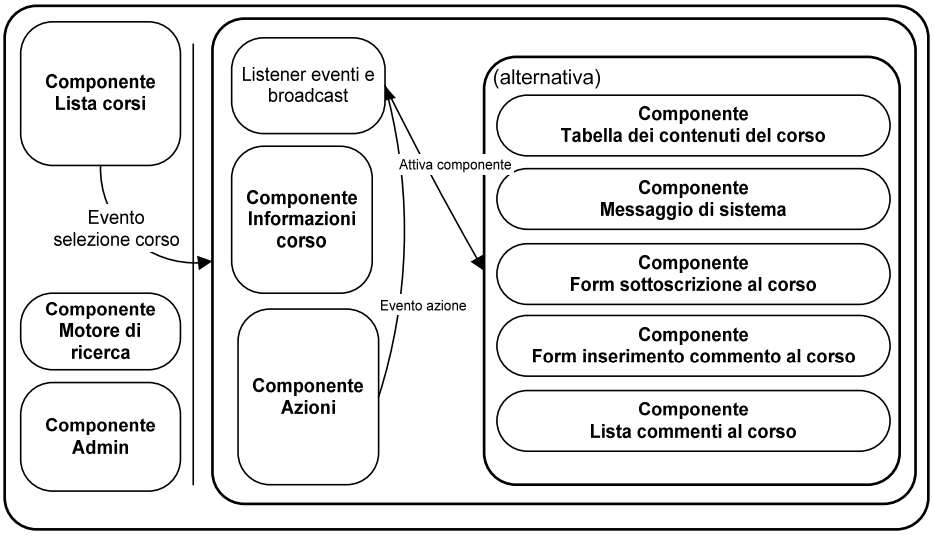


Figura 39: I componenti nell'area (alternativa) sono attivati e visualizzati uno alla volta a seconda dell'evento proveniente dal componente azioni.

- Sistema per il logging delle eccezioni ed errori: quando un componente lancia un messaggio alla pagina contenitore quest'ultima si occupa di generare delle entry nel file di log in modo da tracciare eccezioni ed errori, attività utile soprattutto durante la fase di sviluppo e testing dell'applicazione.
- Sistema per gestire la comunicazione fra i vari componenti: è spesso possibile che i diversi oggetti nella pagina debbano interagire fra loro. E' utile quindi definire una sorta di protocollo per gestire questa comunicazione. Tale sistema è implementato mediante i costrutti `event` e `delegate` e definisce delle classi per l'interscambio di informazioni
- Sistema comune per l'accesso alle informazioni: i componenti devono poter accedere ai servizi offerti dal CMS. Invece di chiamare direttamente tali servizi, sono fornite a livello di pagina delle interfacce di programmazione che permettono al componente di ottenere oggetti trascurando tutti i dettagli e la complessità dell'implementazione delle chiamate ai servizi del CMS.

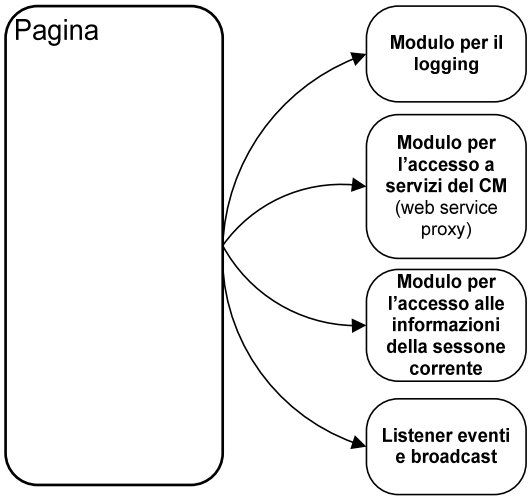


Figura 40: i servizi Modulo per il logging, Modulo per l'accesso a servizi del CM, Modulo per l'accesso alle informazioni della sessione corrente servizi sempre disponibili e vengono istanziati a livello di applicazione mentre il Listener eventi e broadcast è istanziato a livello di pagina.

L'isolamento e la modularizzazione ottenuta mediante questo modello ha però un inconveniente piuttosto consistente: se due componenti devono accedere alle medesime informazioni, probabilmente visualizzandole con due view diverse, l'accesso al CMS, tramite API, sarà multiplo. Ciò comporta, oltre a problemi di efficienza anche problemi di consistenza delle informazioni visualizzate. Infatti potrebbe essere possibile la situazione in cui:

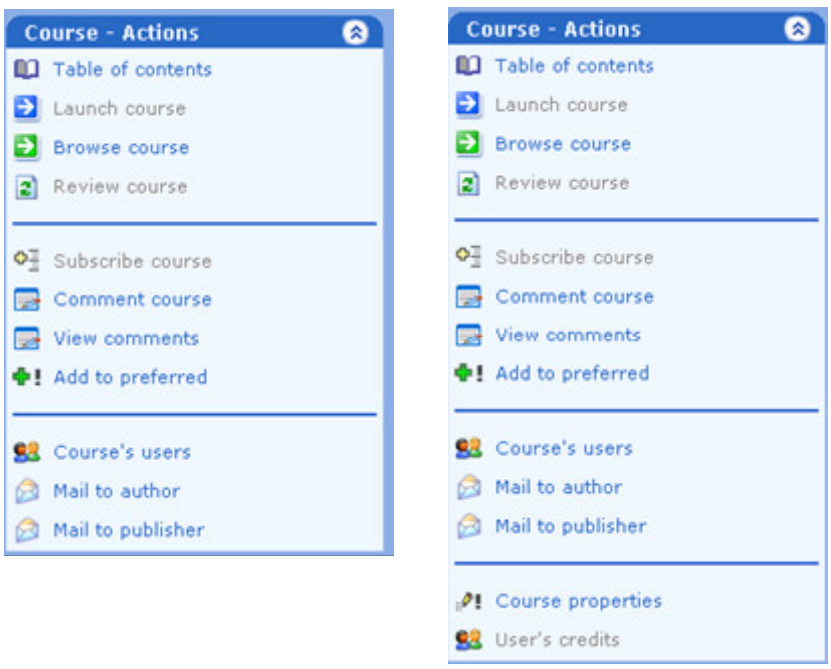
- Il componente A , durante il rendering della pagina, richiede le informazioni I
- Le informazioni I vengono spedite dal CMS al componente A
- Le informazioni I vengono modificate da un altro client e diventano I'.
- I componente B, nella stessa pagina del A e durante la stessa fase di rendering, richiede le informazioni I.
- Le informazioni I' vengono spedite al componente.
- Sullo schermo l'utente vede una view di I e di I'.

Una situazione come quella precedente, in una applicazione come leziNET , è tuttavia sporadica in quanto gli aggiornamenti dei contenuti sono avvenimenti, rispetto alla fruizione, rari. Questi problemi di consistenza non sussistono durante le operazioni di aggiornamento dei dati in quanto la gestione dell'accesso multiplo ai dati in lettura e scrittura è demandata interamente al DBMS.

7.4 Interfaccia utente

L'interfaccia del sistema con la quale l'utente si deve rapportare rappresenta una parte critica dell'applicazione. Pur offrendo funzionalità avanzate un sistema deve essere in grado di ripresentare le informazione in modo semplice e chiaro ma soprattutto deve fornire un meccanismo intuitivo per poter eseguire le diverse operazioni necessarie per poter utilizzare in modo efficiente il sistema. Le moderne interfacce delle applicazioni utilizzano concetti che permettono di astrarre dalla realtà del mondo informatico avvicinandosi alla realtà della vita quotidiana. Si è passati, ad esempio, dal concetto informatico di directory a quello di cartella, proprio perché la cartella, nell'idea comune, è un oggetto che può raccoglierne altri. Esistono infiniti altri esempi come il desktop che è la scrivania, i files che sono diventati documenti etc.

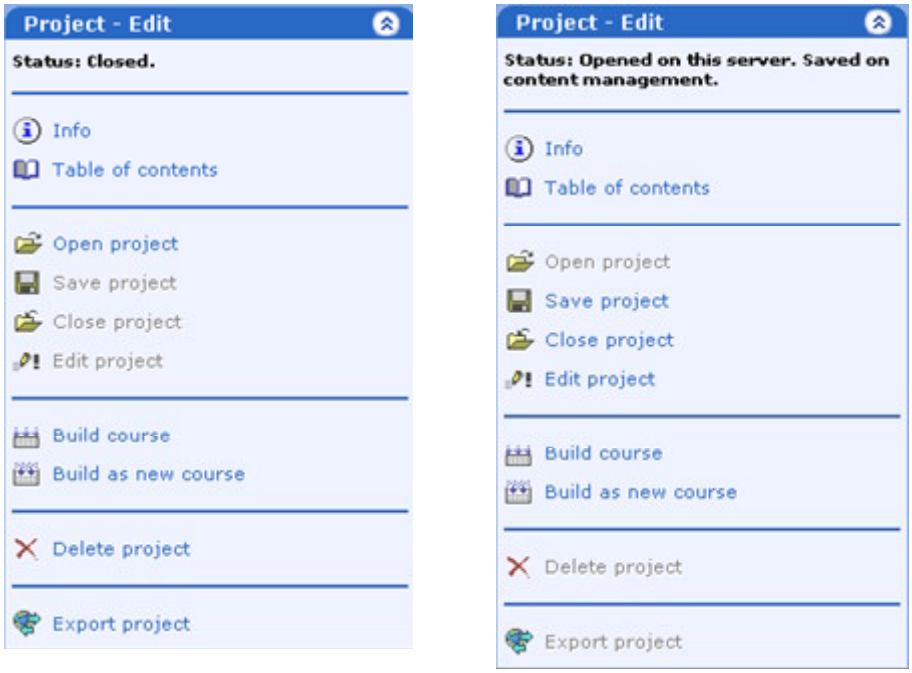
L'interfaccia delle pagine Web dell'applicazione è stata progettata tenendo conto delle precedenti considerazioni. Dal punto di vista degli elementi grafici e della combinazione di colori ci si è ispirati liberamente ad un tema di un prodotto commerciale esistente. Invece dal punto di vista dell'organizzazione dei contenuti, cioè funzionalità offerte e presentazione delle informazioni, la filosofia con cui è stata progettata l'interfaccia è la seguente: se sullo schermo dell'utente sono presenti le informazioni di un oggetto, vicino alla presentazione dei suoi dati, devono essere presenti tutte le azioni che è possibile eseguire con tale oggetto. Si tratta di una sorta di menu contestuale che permette immediatamente all'utente di capire cosa può fare con l'oggetto che sta vedendo. La composizione del menu delle azioni che l'utente può eseguire sull'oggetto è determinato da tre differenti fattori: il primo è naturalmente il tipo di oggetto visualizzato che determina l'insieme di tutte le operazioni eseguibili indipendentemente dallo stato dell'oggetto a dal ruolo dell'utente. Il secondo è lo stato dell'oggetto che filtra eventualmente l'insieme delle azioni determinate dal primo fattore. Il terzo è un ulteriore filtro che riduce ancora, se necessario, l'insieme delle operazioni in base al ruolo che l'utente corrente ricopre. Da punto di vista grafico le diverse azioni, eseguibili sull'oggetto, possono essere efficacemente rappresentate da delle icone munite di etichetta. Se una particolare azione a causa, ad esempio, del ruolo dell'utente, non fosse eseguibile l'icona e l'etichetta non dovrebbero essere visibili. Se invece, pur l'utente avendone la possibilità, non fosse possibile eseguire una certa operazione, l'icona e l'etichetta dovrebbero essere visibili seppur disabilitate. In pratica si tratta di nascondere le operazioni che non sarebbero comunque eseguibili da l'utente a causa de suo ruolo e di visualizzare ma disabilitare le operazioni che non sono eseguibili solo a causa dello stato in cui si trova in quel istante l'oggetto. Questo per evitare di confondere l'utente: se vedesse sempre tutte le operazioni eseguibili, anche quello non appartenenti al suo ruolo, avrebbe una certa quantità di "grafica" sullo schermo inutile. Le azioni disabilitate sono invece da visualizzare poiché l'utente potrebbe trovarsi disorientato non trovando più la voce di una operazione che aveva già visto. In Figura 41 è possibile vedere un esempio di alcuni menu di azioni successivamente filtrati in base al ruolo dell'utente e allo stato dell'oggetto.



a. b.

Figura 41: Medesimo menu filtrato in base al ruolo dell'utente.

Nei menu a. di Figura 42 si hanno le medesime voci del menu b. tranne per le due ultime voci che riguardano la gestione del corso e le prestazioni degli utenti che lo hanno frequentato. E' chiaro quindi che l'utente del menu b. ha un ruolo di Administrator o un ruolo di Creator con diritti di scrittura sul corso selezionato. Le voci aggiuntive di b. essendo delle voci per funzionalità avanzate non vengono nascoste.



a. b.

Figura 42:Menu del progetto di un corso filtrato in base allo stato dell'oggetto, in questo caso un progetto.

Nella nei menu a. e b. di Figura 42 appartengono al medesimo utente, tuttavia nel menu a. lo stato del progetto a cui si riferisce la figura è closed e quindi le voci "Save project", "Close project" e "Edit project" sono disabilitate, e non nascoste. Aprendo il progetto le voci debilitate in a. sono abilitate in b. tuttavia, essendo il progetto aperto, le voci "Delete project" e "Export project" vengono disabilitate.

8 Implementazione

8.1 Applicazione Web Lezi.NET

L'applicazione lezi.NET, come visto nella sezione che descrive l'architettura, ha comportato la realizzazione di due applicazioni Web: la prima per implementare le funzionalità del front tier e la seconda per implementare le funzionalità del middle tier .
Per il back tier si è utilizzato un DBMS esistente.
In questa sezione si parlerà del front tier che realizza tutto il lato di presentazione dell'applicazione fornendo all'utente delle pagine Web tramite le quali è possibile interagire con il sistema.

Considerato che i requisiti impongono che il sistema, ad esempio durante la fruizione, possa identificare l'utente che sta seguendo un corso e possa regolare, in base ad ACL l'accesso degli utenti alle varie risorse , è necessario che l'utente esegua una autenticazione tramite inserimento di un identificativo utente (userID) ed una password. Una volta autenticato il sistema è in grado di modificare dinamicamente l'interfaccia utente in base al ruolo che l'utente ricopre. Ad esempio, se si tratta di un utente normale, quindi appartenente al solo gruppo users, l'interfaccia nasconderà tutti i pulsanti relativi all'amministrazione del sito e alla creazione dei progetti di corsi.
Questo comporta che se l'utente fosse a conoscenza dell'URL di una pagina interna al sito, raggiungibile solo ad autenticazione avvenuta, il sistema dovrebbe bloccare l'elaborazione della pagina inviando un messaggio di errore oppure redirezionando l'utente alla pagina di login. Il framework ASP.NET fornisce un insieme di classi per poter gestire agevolmente situazioni di questo tipo. Queste classi utilizzate in combinazione il metodo di autenticazione "Forms" , permettono di rimandare l'utente alla pagina di login se quest'ultimo non si è ancora autenticato. Il sistema di autenticazione implementato utilizza tre pagine: una pagina alla quale l'utente vuole arrivare , ad esempio Default.aspx la pagina di sistema Global.asax (Vedi sezione 4.4), e la pagina Login.aspx. I passaggi effettuati sono i seguenti:

- L'utente richiede la pagina Default.aspx
- Il sistema di autenticazione Forms redireziona l'utente alla pagina Login.aspx memorizzando la pagina richiesta originalmente.
- La pagina Login.aspx non ha abilitata le gestione della sessione.
- L'utente esegue la login inserendo userid e password.
- Il sistema verifica le credenziali fornite, rigirando la verifica al lezi.NETCMS .
- Se la password è corretta, il sistema riporta l'utente alla pagina richiesta in precedenza, in questo caso Default.aspx.
- Siccome questa pagina ha abilitata la gestione della sessione, verrà lanciato dal framework un evento che provocherà la chiamata di un metodo di Global.asax, che provvederà ad inizializzare le variabili di sessione e fornirà all'utente un token (un hash calcolato in base al nome dell'utente, alla userid, alla password a alla data e ora corrente di sistema) che sarà utilizzato d'ora in poi in modo trasparente all'utente per tutte le operazioni in cui è richiesta l'autenticazione per una successiva autorizzazione.

Una soluzione di questo tipo ha i seguenti vantaggi:

- Utilizza un sistema, il sistema Forms, che, insieme ad un set di classi, permette di implementare un solido sistema di autenticazione. Implementare nuovamente software che fornisce funzionalità già esistenti sarebbe inutile e soprattutto aumenterebbe le linee di codice scritto, aumentato così la possibile presenza di difetti (bugs).
- Il sistema utilizzato ha evitato di dover inserire in ogni pagina un controllo per verificare l'avvenuta autenticazione e le righe di codice necessarie per l'inizializzazione delle variabili di sessione. La logica che si occupa di queste operazioni è presente solo nel Global.asax.

Effettuata l'autenticazione, l'utente ha accesso all'applicazione la cui mappa statica è schematizzata in Figura 43.

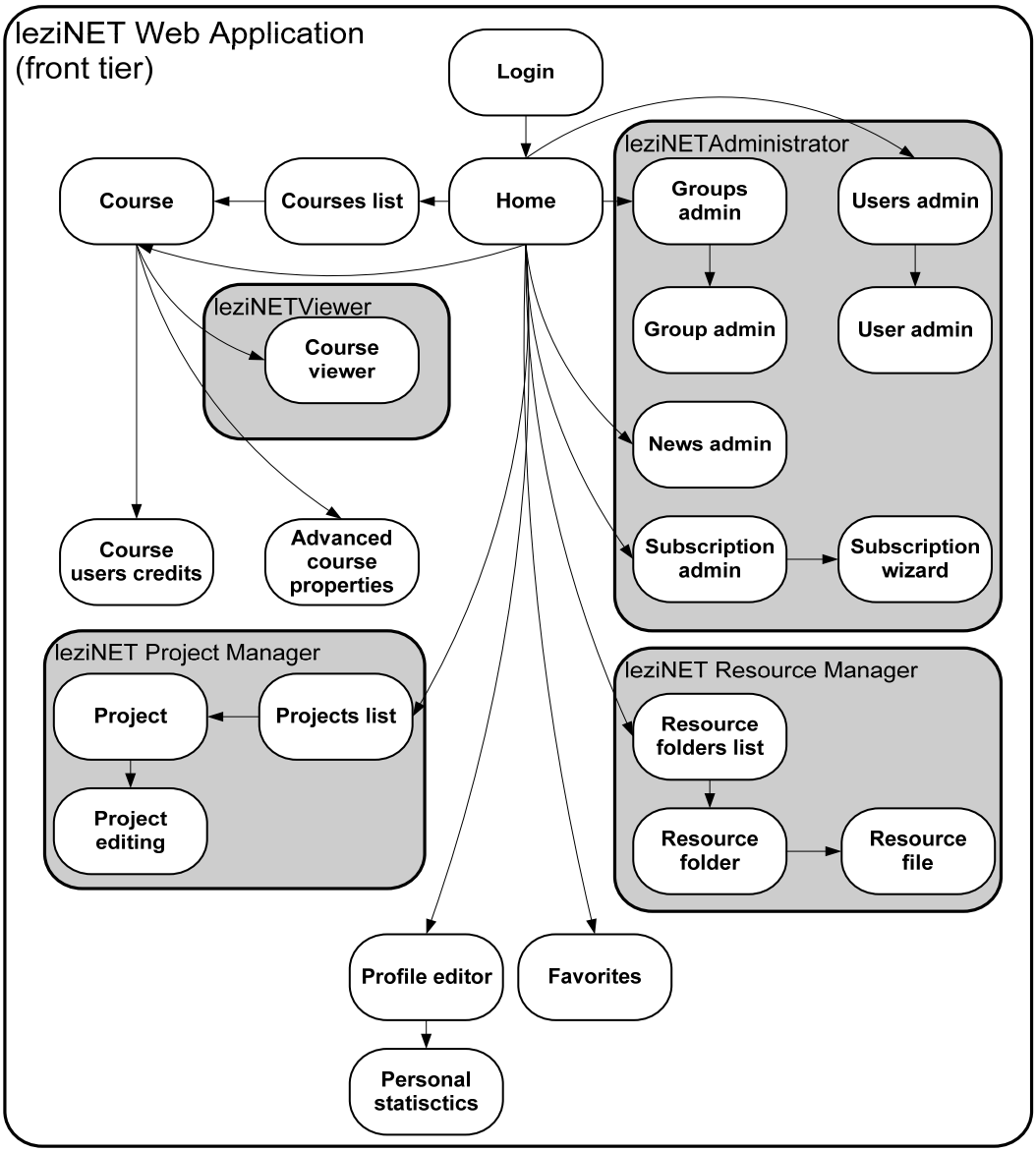


Figura 43: mappa sintetica (parziale) dell'applicazione. Ogni quadratino rappresenta una pagina, nella quale sono presenti una o più funzionalità realizzate mediante l'utilizzo di componenti.

Dietro la mappa di Figura 43 esiste, a causa della gran quantità di pagine necessarie per realizzare le funzionalità richieste, una organizzazione della posizione fisica, rivista spesso

durante l'implementazione, delle pagine aspx, delle librerie, delle directory temporanee dei files di configurazione e dei files di log. Una disposizione di questo tipo non ha fini funzionali ma solo organizzativi atti a semplificare la fase di sviluppo ma soprattutto la fase di testing e manutenzione. La figura 44 riassume alcuni dettagli riguardanti tale strutturazione.

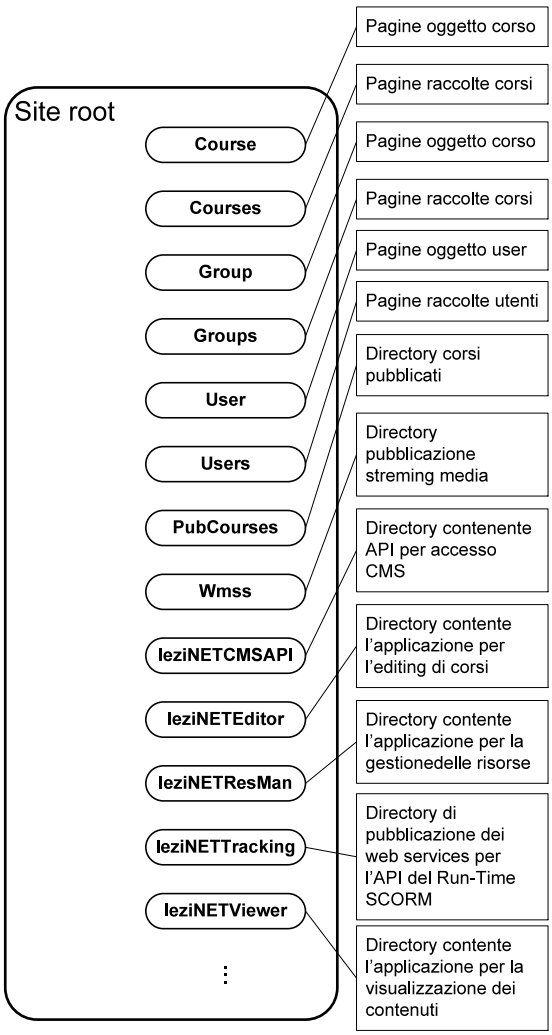


Figura 44: Struttura parziale che schematizza l'organizzazione "fisica" interna dell'applicazione di front leziNET. Con il termine "pagine raccolte…" si intendono quelle pagina in cui sono visualizzate delle view, solitamente tabellari, di insiemi di oggetti dello stesso tipo.

Le sezioni in grigio di Figura 43 che rappresentano sotto applicazioni facenti parte dell'applicazione Web principale, verranno in seguito descritte in dettaglio.

8.1.1 Lezi.NET Viewer

L'ambiente di visualizzazione dei contenuti per poter essere IMS/SCORM compatibile deve fornire all'utente gli strumenti per l'interazione con i contenuti e l'ambiente di run-time per permettere ai contenuti di interagire con il tracking service dell'LMS. Nonostante la comunicazione fra Web browser e Web server sia di tipo bidirezionale, la trasmissione di informazioni dal browser al server può avvenire solo tramite l'operazione di POST del protocollo http. Ciò provoca la costruzione di una request che viene spedita al server il quale produrrà un response. Questa serie di operazioni causa il caricamento di una nuova pagina nel browser. L'interazione che il run-time deve poter avere con l'LMS deve essere

completamente trasparente all'utente il quale deve, durante questa interazione, poter continuare a fruire con continuità della pagina corrente. E' necessario quindi un nuovo canale di comunicazione, il canale di tracking, che si affianca al canale di distribuzione. Attraverso il canale di distribuzione, implementato mediante il normale protocollo di comunicazione fra browser e Web server, il browser chiede ed ottiene le pagine dei contenuti, mentre attraverso il canale di tracking, implementato come si vedrà in seguito attraverso Web services, avviene tutta la comunicazione fra il run-time (client side) e l'LMS.

Riassumendo:

1 – Il browser chiede il contenuto da fruire.
2 – Il servizio di distribuzione dell'LMS fornisce il contenuto
3 – Il browser, che ospita il run-time, manda in esecuzione il contenuto
4 – Durante la fruizione del contenuto questo ultimo interagisce in modo trasparente all'utente con il run-time
5 – Il run-time interagisce, tramite il canale di tracking, con l'LMS comunicando dati ed eventi relativi alla fruizione della pagina del contenuto da parte dell'utente.

Tenendo conto di questi diversi aspetti,. l'ambiente di esecuzione è costituito da una pagina Web composta da diversi frames ognuno dei quali contiene codice JScript per realizzare tutte le funzionalità richieste. Vedi Figura 45.

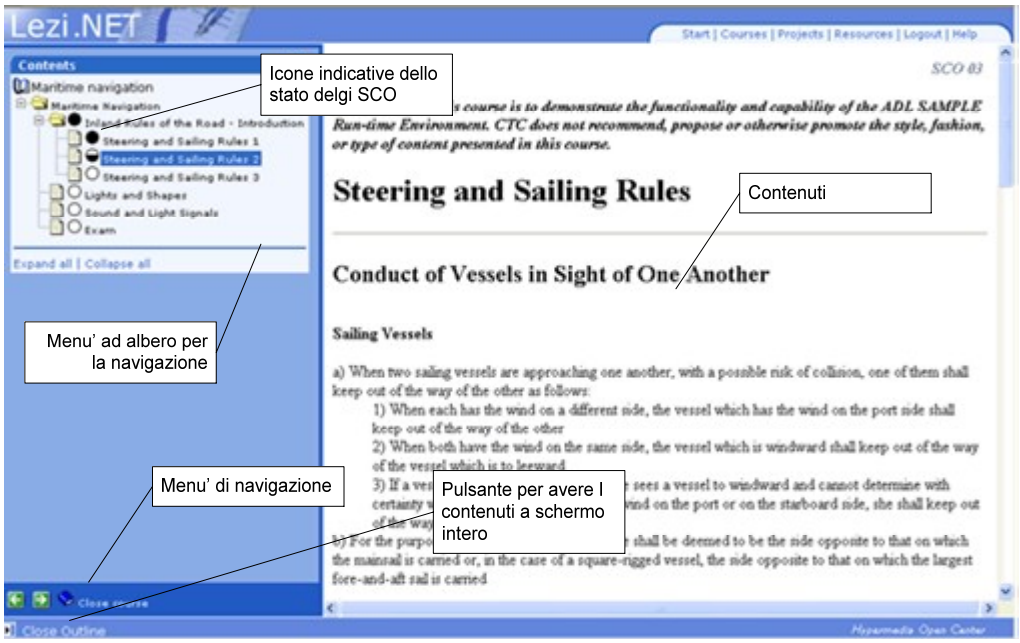


Figura 45: layout del viewer lezi.NET

Prima di parlare delle varie parti e dei vari frames della pagina è necessario prima parlare di tre componenti, tre "prototype" JScript, che permettono di raggruppare le funzionalità organizzando il codice.

SBMenu:
Un insieme di classi JScript che forniscono delle API per la creazione di menu ad albero client side, cioè il codice che gestisce le operazioni di selezione, espansione, collassamento

etc è eseguito dal browser. Il codice di questa classe risiede su un file separato sbmenu.js che viene importato nella pagina in modo da separare il codice di presentazione dalla business logic per il controllo del menu e permettere un eventuale riutilizzo dell'API in più pagine.

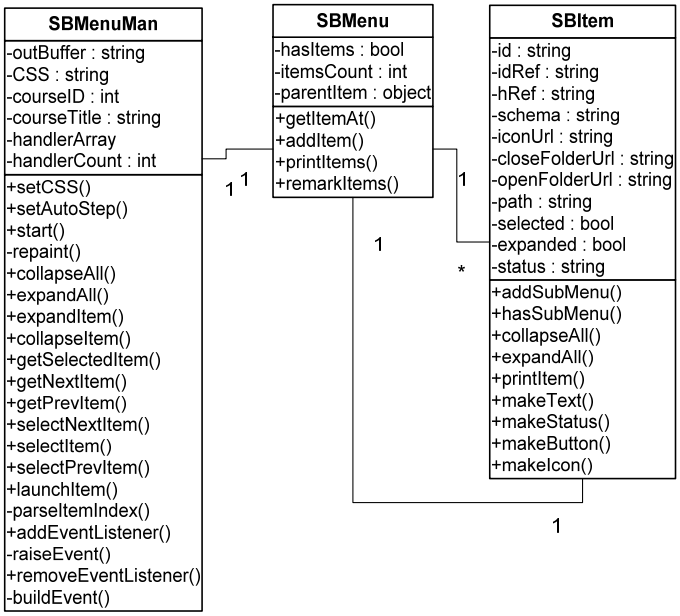


Figura 46: diagramma delle classi della API SBMenu per poter generare un menu ad albero in JScript client-side.

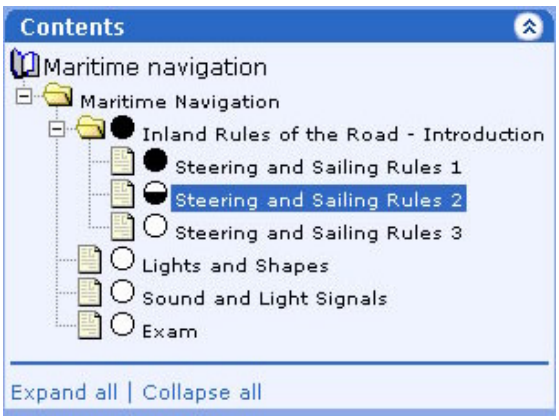


Figura 47: menú ad albero costruibile con le API SBMenu.

RteApi:
Classe JScript che implementa l'API richiesta dal run-time nel file rteapi.js. Questa classe fornisce l'interfaccia di comunicazione fra in contenuti SCORM e l'LMS. Siccome il tracking service del LMS in leziNET è implementato mediante Web service è necessario che, una volta chiamato un metodo da parte de contenuto, venga costruito un pacchetto XML SOAP in modo da invocare il metodo corrispondente sul tracking service. Ad esempio, se è invocato il metodo LMSInitialize() utilizzando l'oggetto api.LMSIntialize(), il corpo di tale metodo costruirà un messaggio XML-SOAP che invocherà il corrispondente Web service di nome LMSInitialize passando correttamente i parametri forniti dal contenuto.

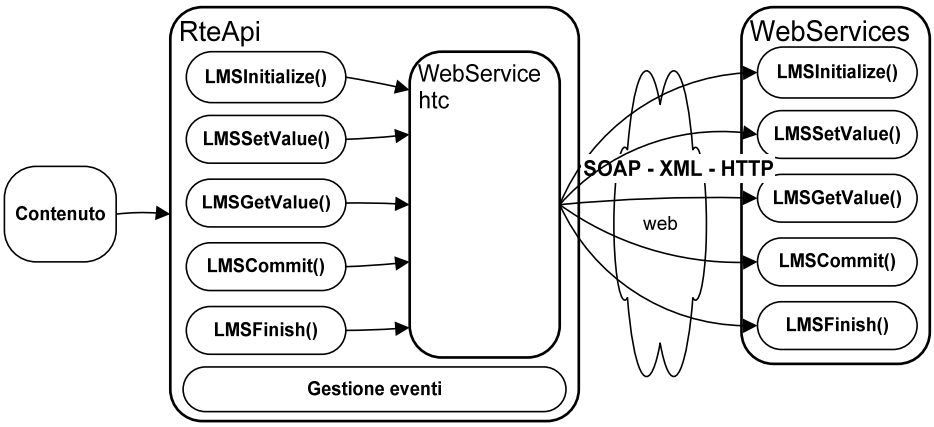


Figura 48: schema di interazione fra contenuto, api del run-time SCORM e mappatura delle chiamate locali su rispettivi Web services pubblicati sull'LMS.

WebServiceHTC:
Classe che fornisce l'API per la creazione, invio e ricezioni di messaggi XML-SOAP. Tale classe è realizzata mediante DHTML behaviors e permette la spedizione di messaggi sia in modo asincrono che modo sincrono. Per le necessitá implementative del run-time specificato da SCORM è utilizzata solo la modalità sincrona in quanto la comunicazione con l'LMS deve essere bloccante e l'esecuzione del contenuto non può procedere senza che una notifica di avvenuta comunicazione sia stata ricevuta.

MenuNav:
Classe che implementa un piccolo menu di navigazione composto di tre pulsanti: next, previous e close.

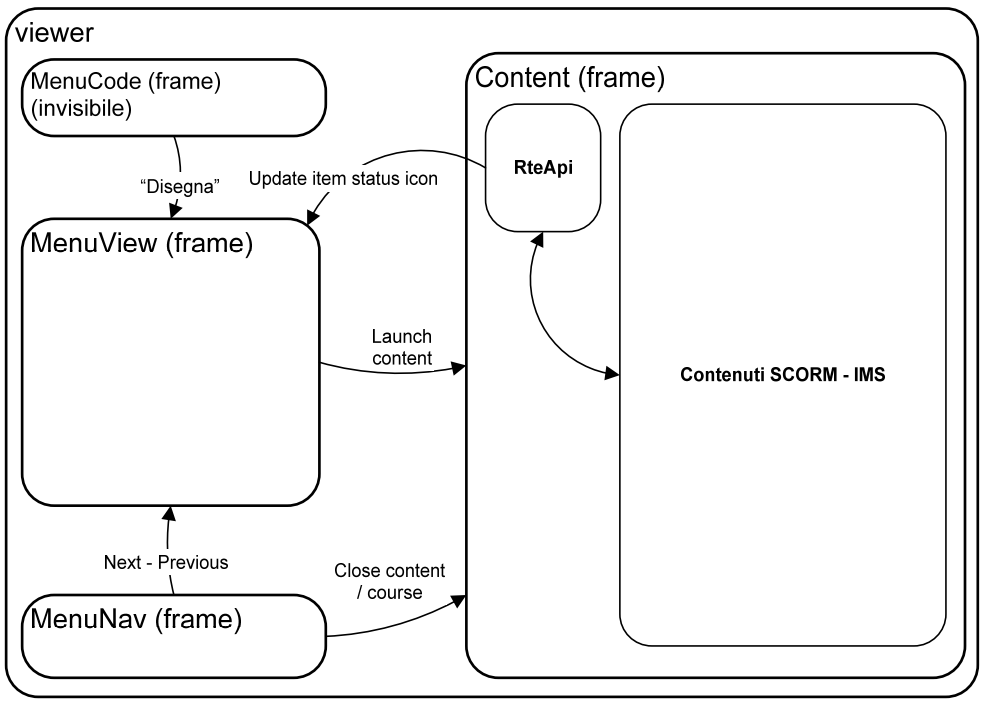


Figura 49: posizione indicativa dei frames che compongono il viewer lezi.NET. Con le frecce sono indicati i messaggi fra i vari oggetti dei frames. Le spedizioni di tali messaggi sono implementati mediante un sistema ad eventi.

Frames del viewer:

Un frame invisibile detto **MenuCode** ospita una pagina attiva ASP.NET sulla quale viene creato dinamicamente codice JScript che servirà a sua volta per la costruzione del menu ad albero (client side) contenente l'indice dei contenuti. Il motivo per cui ho usato un frame invisibile che scrive codice DHTML in uno visibile è che lo stato del menu viene conservato nel frame invisibile che non viene mai ricaricato, mentre il frame visibile, che viene ridisegnato ad ogni operazione (ad esempio espansione di un ramo) perde e ricarica il proprio stato ogni volta. Con maggiore dettaglio:

Server side:
Eseguendo il parsing dell'IMS manifest (sezione organization) si crea codice JScript per la costruzione dell'albero client side utilizzando l'API, precedentemente descritta, SBMenu. Ad esempio:

```
// Generato server side
//*********************
menuMan.s_courseTitle='Maritime navigation';menuMan.setAutoStep('True')
var item1 = new SBItem('Maritime Navigation','B1','','','False','ADL SCORM','');
var menu1 = new SBMenu();
item1.addSubMenu(menu1);
var item2 = new SBItem('Inland Rules of the Road -
Introduction','S1','sco_1','sco01.htm','True','ADL SCORM','completed');
var menu2 = new SBMenu();
item2.addSubMenu(menu2);
var item3 = new SBItem('Steering and Sailing Rules
1','S2','sco_2','sco02.htm','True','ADL SCORM','completed');
menu2.addItem(item3);
var item4 = new SBItem('Steering and Sailing Rules
2','S3','sco_3','sco03.htm','True','ADL SCORM','not attempted');
menu2.addItem(item4);
var item5 = new SBItem('Steering and Sailing Rules
3','S4','sco_4','sco04.htm','True','ADL SCORM','not attempted');
menu2.addItem(item5);
menu1.addItem(item2);
var item6 = new SBItem('Lights and Shapes','S5','sco_5','sco05.htm','True','ADL
SCORM','not attempted');
menu1.addItem(item6);
var item7 = new SBItem('Sound and Light
Signals','S6','sco_6','sco06.htm','True','ADL SCORM','not attempted');
menu1.addItem(item7);
var item8 = new SBItem('Exam','S7','sco_7','sco07.htm','True','ADL SCORM','not
attempted');
menu1.addItem(item8);
menu.addItem(item1);
//*********************
```

Il precedente codice JScript, generato server-side mediante , permette la creazione del menu in Figura 47.

Client side:
Il codice precedente, presente nella pagina spedita al browser dal server, una volta mandato in esecuzione istanzia i diversi oggetti che compongono il menu. Chiamando il metodo paint() sulla sul menuItem radice viene scritto (document.write(..)) codice DHTML che interpretato dal browser disegna, in un altro frame chiamato **MenuView**, il menu dei contenuti.Vedi Figura 49.

Il frame centrale, chiamato **Contents**, fornisce l'oggetto "api" che costituisce l'interfaccia fra i contenuti ed il run-time. La pagina ospitata in questo frame, mandata in esecuzione, provvede ad instanziare l'oggetto "api" del run-time utilizzando RteApi.
Infine un frame, posizionato in basso a sinistra che, istanziando un oggetto **MenuNav** fornisce i pulsanti previous, next e close.

lezi.NET fornisce lo stesso ambiente d'esecuzione sia per i contenuti di tipo IMS che quelli SCORM. Naturalmente quelli IMS non faranno uso dell'RteApi. In questo modo non è stato necessario diversificare l'ambiente di esecuzione in base al tipo di contenuto. I corsi IMS semplicemente non utilizzano un servizio che viene comunque messa a disposizione dal viewer.

Un aspetto non esplicitamente considerato nella documentazione del run-time SCORM riguarda il fatto che un contenuto, uno SCO, dovrebbe chiamare, prima di essere interrotto nella propria esecuzione, i metodi LMSCommit e LMSFinish. Tuttavia, sostituendo lo SCO corrente con il nuovo capitava spesso che il "vecchio" SCO non riuscisse a chiamare in tempo LMSCommit e LMSFinish prima che il "nuovo" SCO chiamasse LMSInitialize. Questo problema di sincronizzazione, dipendente anche da ritardi di rete non controllabili, è stato risolto introducendo una pagina di supporto che è caricata fra lo SCO vecchi e quello nuovo. Tale pagina esegue le operazioni:

1 – Causa il lancio dell'evento unload da parte del browser permettendo allo SCO di accorgersi che il contento sta per essere fermato. Lo SCO chiama i metodi LMSCommit e LMSFinish.
2 – Visualizza il messaggio "Loading lesson…" all'utente notificando l'operazione di caricamento di un nuovo SCO.
3 – Contiene codice JScript scritto dinamicamente che esegue un redirect client-side verso la pagina attiva, launcher.aspx, server-side, che eseguirà il caricamento del nuovo SCO.

La pagina launcher.aspx, prima di mandare in esecuzione il nuovo SCO verifica che lo SCO vecchio abbia terminato correttamente la propria esecuzione. Se tale verifica ha esito negativo aspetta qualche secondo usando Thread.sleep(1000) per evitare busy waiting. Se l'esito è ancora negativo aspetta nuovamente fino ad un massimo di dieci volte un secondo dopodiché, il sistema assume che a causa di problemi di rete o dello SCO quest'ultimo non termini correttamente, resetta la macchina a stati del run-time e manda in esecuzione lo SCO nuovo.

Per gestire le interazioni fra gli oggetti posizionati nei diversi frames è stato necessario istituire un sistema ad eventi. Per questo motivo alle varie classi, SBMenu, LMSRTE, MenuNav sono stati aggiunti i metodi:

```
public addEventListener()
public removeEventListener()
private buildMessage()
private raiseEvent()
…
```

In questo modo se dovesse accadere qualcosa in un frame e altri oggetti, anche in altri frames, fossero interessati a quell'evento basterebbe registrarli all'oggetto che lancia quel tipo particolare tipo di evento.
Ad esempio:

1 – L'utente preme il pulsante Next nel frame MenuNav. Vedi Figura 49.

2 – L'istanza dell'oggetto della classe MenuNav[35] costruisce e lancia l'evento "premuto next".
3 – A tutti gli oggetti in precedenza iscritti all'evento presso menuNav, viene spedito un messaggio indicante il tipo di evento.
4 – L'oggetto sBMenuNav, deve, ricevendo questo tipo di messaggio, spostare la selezione corrente sull'item successivo.
5 – Quest'azione causa la spedizione di un messaggio "selected item changed" a tutti gli oggetti interessati a tale tipo d'evento.
6 – Contents ricevendo il messaggio "selected item changed" forzerà l'unload della pagina del contenuto corrente causando la chiamata, da parte del contenuto stesso a LMSCommit e LMSFinish.
7 – Inoltre sempre Contents manderà in esecuzione, chiamando la pagina attiva launcher.aspx il contenuto il cui id era indicato nel messaggio spedito da MenuCode.
8 – La pagina, con il nuovo contenuto chiamerà il metodo del run-time LMSInitialize.
9 – Quest'ultima azione causa la spedizione di un messaggio a MenuCode, il quale si era naturalmente iscritto a tale messaggio, per modificare l'icona che indica lo stato del contenuto appena partito.
10 …

Questa sequenza d'azioni, che descrive solo parte di un possibile scenario, permette di capire quanto sia ampio il sistema ad eventi che è stato necessario implementare per poter realizzare un viewer conforme con le specifiche SCORM 1.2. Inoltre, causa la quantità d'interazione fra i vari oggetti e i differenti tipi di messaggi che gli oggetti si possono scambiare, si è stati obbligati ad utilizzare un design pattern molto potente il Model – View – Controller (Vedi MVC), utilizzato spesso nelle applicazione attuali per l'implementazione di interfacce grafiche complesse in cui spesso lo stesso oggetto, come nel caso di lezi.NETViewer, funge sia da viewer che da controller.

8.1.2 Lezi.NET Resource manager

Il modulo per la l'immagazzinamento e la gestione delle risorse permette all'utente di caricare sul sistema files d'ogni tipo, che potranno essere poi utilizzati per costruire i corsi. Il resource manager consta principalmente di alcune pagine Web e di componenti software per l'upload, la compressione ed il trasferimento dei dati. L'interfaccia utente che è una parte dell'applicazione del front tier fornisce un ambiente simile ad un comune file manager E' infatti possibile organizzare i propri files in cartelle, spostarli, copiarli, incollarli e visualizzarne le proprietà . Lo scopo è quello di fornire un ambiente con cui l'utente creatore di corsi fosse il più familiare possibile.

[35] *Il nome del frame spesso coincide con il nome della classe principale dell'oggetto Jscript che il frame stesso istanzia all'avvio e poi ospita durante l'esecuzione. Ad esempio, il frame MenuNav istanzia un oggetto JScript di classe MenuNav creando l'oggetto menuNav. Frame, classe, API e istanza, pur avendo spesso lo stesso nome non sono da confondere in quanto oggetti di tipo diverso..*

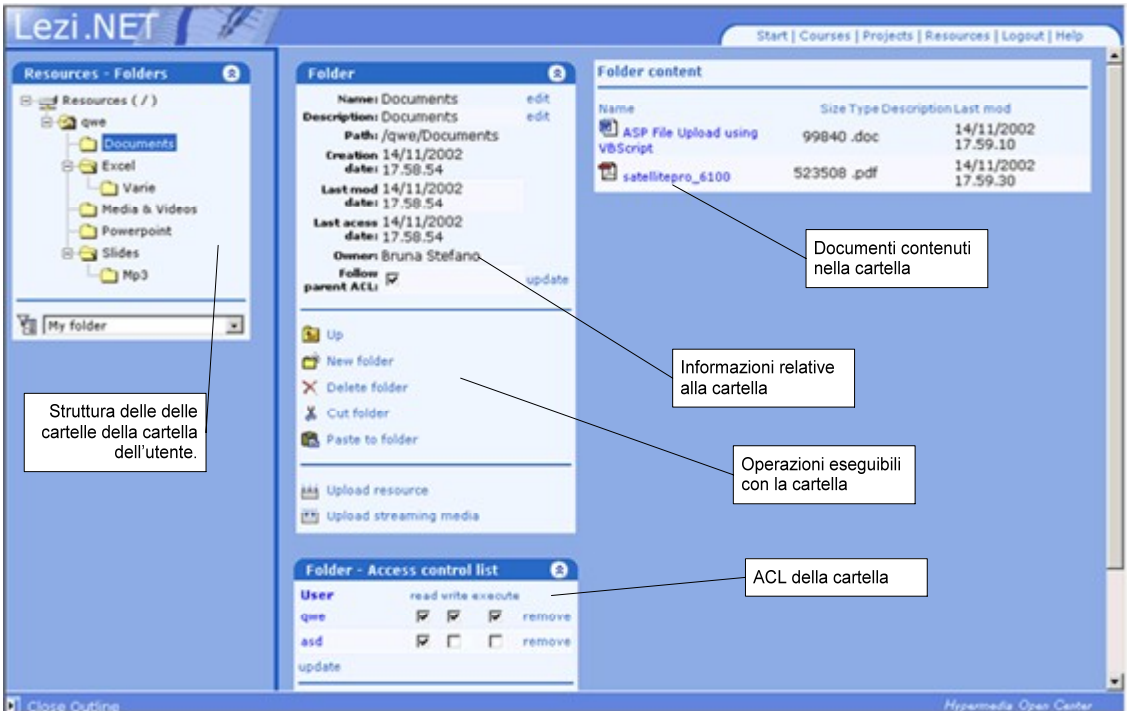


Figura 50:Layout della pagina della cartella documenti.

E' possibile eseguire due differenti operazioni di upload differenti. Uno è dedicato ai file di piccole dimensioni, tipicamente sotto i 10 megabytes ed uno dedicato ai files multimediali di grandi dimensioni come filmati ad alta qualità e/o lunga durata. I primo tipo di upload è stato implementato utilizzando il software fornito dal framerwork .NET. Mediante l'attributo type di una form html è possibile indicare al Web server che si vuole eseguire un upload invece che spedire i dati della form. Server side è fornito un oggetto, raggiungibile mediante Request.Files, che contiene un riferimento allo stream di dati che permette il salvataggio dei dati sul file system del Web server. Questa soluzione implementativa pur essendo semplice e velocemente utilizzabile ha un grande limite che diventa molto evidente quando le dimensioni dei files di cui si esegue l'upload aumentano. Eseguendo, ad esempio, l'upload di un filmato da 100 Mb, durante lo stream, questo viene completamente portato in memoria nella zona dati del processo del Web server. Quando il filmato è salvato durante l'esecuzione della pagina aspx l'eseguibile torna alle dimensioni precedenti l'operazione (sempre che il garbage collector operi in modo efficiente). Se tuttavia l'operazione precedente fosse eseguita contemporaneamente da più utenti, per evitare pesanti rallentamenti, bisognerebbe fornire il Web server di una quantità molto grande di memoria solo per supportare alcuni upload contemporanei. Inoltre durante l'installazione del lato server del framework .NET viene di default installata nel sistema una policy che per motivi di efficienza e sicurezza termina il processo del Web server quando questo supera 80% della dimensione della memoria di sistema. Poi, naturalmente, il processo è rilanciato subito. Per questi motivi è stato necessario sviluppare un sistema più efficiente per eseguire il caricamento nel sistema di files di grandi dimensioni. Fortunatamente il Web server .NET è largamente e facilmente modificabile. E' infatti possibile intercettare le request dei client e identificando quelle degli upload mediante, ad esempio, di una particolare stringa nella queryString della richiesta, é possibile gestire direttamente il flusso di dati che arriva dal client salvandolo direttamente sul file system utilizzando eventualmente una decina di megabytes come cache memory per rendere ancora più efficiente e veloce l'operazione. In Figura 51 è possibile vedere le operazioni eseguite dalla coppia client-server durante le operazioni di upload di un file di grandi dimensioni.

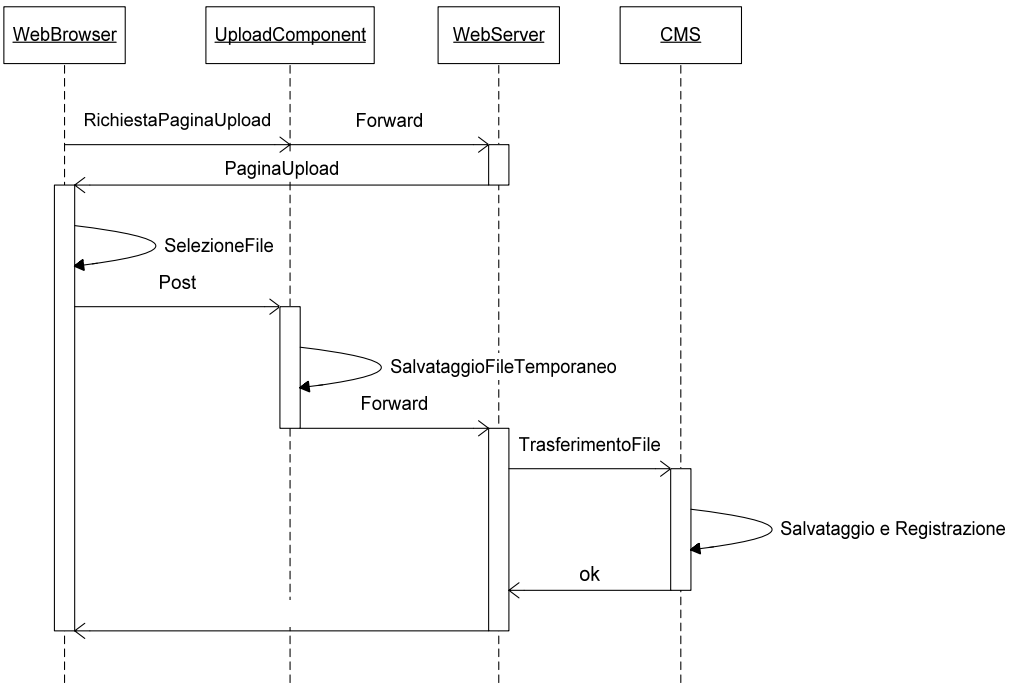


Figura 51: Sequenza delle operazioni durante un salvataggio di un file utilizzando l'upload per i files di grandi dimensioni. Si noti come il componente per l'upload intercetti e trasmetta le richieste del client a meno che queste non siano un post della form di upload.

8.1.3 Lezi.NET Editor

Lezi.NET fornisce un editor che permette di creare corsi IMS e SCORM. Ogni corso ha una struttura ad albero virtualmente infinita sia in profondità che estensione. Ogni elemento dell'albero può essere una foglia oppure un contenitore d'altri elementi. Lezi.NET editor propone tre tipi d'elementi. Un elemento contenitore privo di contenuto (folder template), in pratica una specie di cartella con fini organizzativi, un elemento in cui è possibile pubblicare testo formattato (text template), ed un elemento (multimedia template) che presenta i contenuti mediante una interfaccia composta da quattro sottofinestre: in alto a sinistra un filmato o, eventualmente, una traccia audio, in alto a destra uno slideshow sincronizzato con il flusso del filmato, in basso a sinistra la lista dei capitoli del filmato (markers) e in basso a destra una lista di documenti e hyperlinks relativi ai contenuti (Vedi Figura 52).

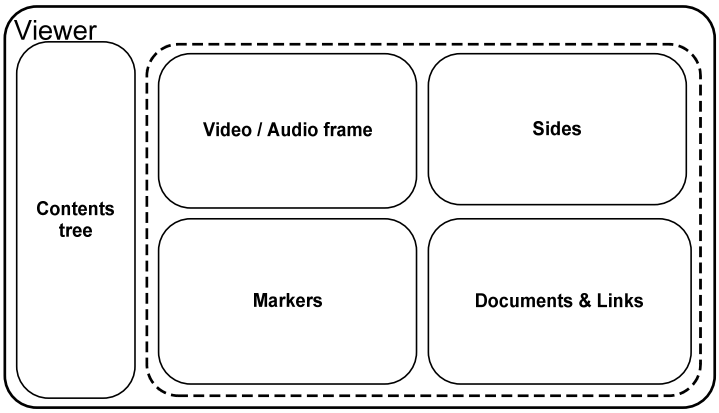


Figura 52: Layout di sezione di un corso producibile con l'editor di lezi.NET. Ad ogni voce presente nel contents tree corrisponde una sezione (area tratteggiata).

La sincronia fra filmato, slides e markers corrente é implementata mediante un sistema ad eventi client side nella pagina html che ospita i contenuti. La pagina, come già detto, è suddivisa in quattro frames. Il frame contenente l'oggetto "player" chiede a quest'ultimo la lista dei marker del filmato corrente per creare dinamicamente la lista dei capitoli del filmato nel frame Marker di Figura 52, cattura inoltre gli eventi provenienti player stesso.

Esempio 14: Codice JScript per l'estrazione dei markers, attraverso l'oggetto player, dal filmato.

```
function InitializeMarkers ()
{
    var iMax, x;

            parent.tocframe.location.reload();
            parent.tocframe.document.write("<html>");
            parent.tocframe.document.write("<style type='text/css'>HR{COLOR:
#316ac5;}</style>");
            parent.tocframe.document.write("<head>");
            parent.tocframe.document.write("</head>");
            parent.tocframe.document.write("<body>");
            parent.tocframe.document.write("<table width='100%'><tr><td>");
            parent.tocframe.document.write("<div><font face='Verdana'
size='2'><b>Contents</b></font></div>");
            parent.tocframe.document.write("<hr>");
            if ((iMax = document.NSPlay.MarkerCount) > 0) {
                  for (x = 1; x <= iMax; x++)
                      parent.tocframe.document.write("<div><font face='Verdana'
size='2'><A href='javascript:parent.playerframe.OnMarkerClicked(" + x + ")'>" +
document.NSPlay.GetMarkerName(x) + "</A></font><BR></div>");

            }
            else
            {
                  parent.tocframe.document.write("No media index");
            }
            parent.tocframe.document.write("<hr>");
            parent.tocframe.document.write("</td></tr></table>");
            parent.tocframe.document.write("</body>");
            parent.tocframe.document.write("<html>")

}
```

Il filmato in streaming eseguito dal player, contiene oltre alla traccia video e audio, una traccia detta "script" nella quale sono stati distribuiti nel tempo i comandi di script sotto forma di nome, comando e parametro. Quando l'esecuzione del filmato incontra un comando di script il player lancerà l'evento corrispondente. E' possibile inserire i comandi

di script in un filmato Windows Media mediante programmi di editing video oppure tramite il Windows Media Indexer contenuto nel Resource Kit di Windows Media.
I comandi di script possono essere di due tipi: “marker hit” e “slide”. Quando il player incontra un comando marker hit lancia un evento che viene catturato dalla pagina che aggiorna la posizione corrente nel frame Marker evidenziando il marker raggiunto.
Ad esempio un filmato può contenere tre marker (Introduzione, Requisiti e Progetto) e n eventi con nome “slide” e parametri numerati da 1 a n.
Al caricamento della pagina, il player, interrogando il filmato costruisce la lista nel frame Markers:

- Introduzione
- Requisiti
- Progetto

Cliccando su una voce, viene lanciato dalla pagina l’evento “marker clicked” che viene catturato dal player che sposterà di conseguenza l’esecuzione dello stream nel punto corrispondente.

Esempio 15: Codice JScript per lo spostamento nel filmato cliccando su una voce.

```
function OnMarkerClicked (marker)
{
    document.NSPlay.CurrentMarker = marker;
}
```

Quando il player incontra un comando di script “slide” con parametro n, la pagina carica nel frame slide la slide corrispondente.

Esempio 16: Codice JScript per la gestione dell’evento slide.

```
<script for="NSPlay" event="ScriptCommand(sType, sParam)" language="JScript">
                    //<!--

                    if (sType == "slide")
                    {
                           parent.slideframe.location.replace(slides[sParam]);
                    }

                    // --></script>
```

Una struttura dati contenente il binding numero slide – URL slide è memorizzata precedentemente nella pagina stessa durante la sua creazione con il leziNET Editor.

Esempio 17: Codice JSript per il caricamento delle slides.

```
<script identifier="slideload" language="javascript">// <!--
var slides = new Array();
slides[1] = 'slide_0.jpg';
slides[2] = 'slide_1.jpg';
slides[3] = 'slide_2.jpg';
slides[4] = 'slide_3.jpg';
// -->
</script>
```

Il fatto di memorizzare i comandi per gestire la sincronia filmato - slide – markers all’interno del filmato sembra essere una soluzione poco flessibile, infatti dovendo modificare, ad esempio, la posizione temporale di una slide, è necessario editare

nuovamente il filmato. Tuttavia è una soluzione obbligata in quanto deve essere necessariamente il filmato a "guidare" il flusso di informazioni. Infatti potenziali ritardi di rete durante lo streaming causerebbero una mancata sincronia con le slide nel caso quest'ultime avessero una temporizzazione separata, ad esempio memorizzata localmente in strutture dati JScript all'interno della pagina.

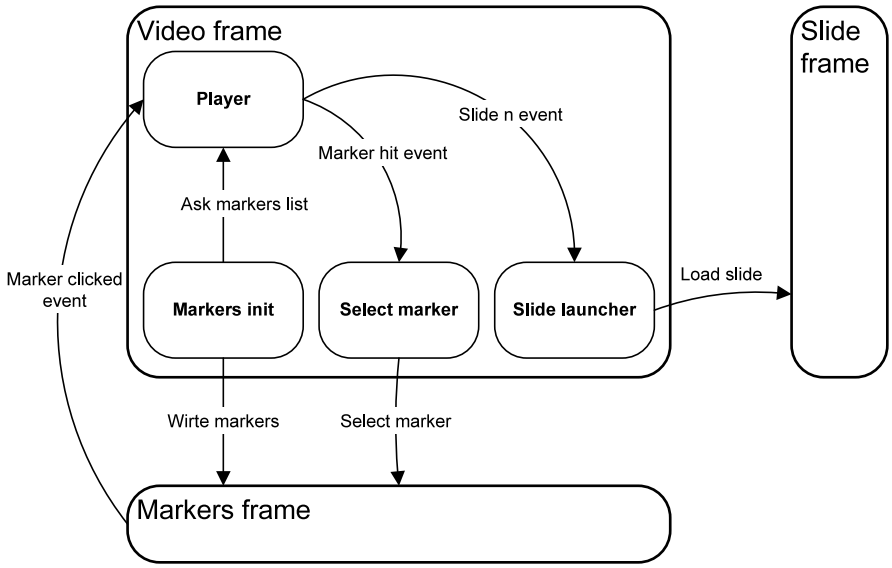


Figura 53: Sistema ad eventi client side per gestione sicronizazione filmato - slide - markers.

Tutti i tre tipi di elementi folder template, text template e multimedia template possono essere contenitori di altri elementi.
L'interfaccia di lezi.NET editor permette facilmente di utilizzare, per la costruzione dei vari elementi, tutti i contenuti presenti nell'archivio personale dell'utente creatore. Ad esempio, per la costruzione di un elemento mediante multimedia template, si dovrà utilizzare un filmato, delle slides ed eventualmente dei documenti o degli hyperlinks relativi ai contenuti. I dati utilizzati per la creazione del corso vengono copiati rimanendo così disponibili nell'archivio dell'utente per una loro futura utilizzazione durante la creazione di altri corsi.
I file video caricati nel sistema come streaming media non vengono fisicamente copiati nel package del progetto ma viene creato un riferimento allo streaming server e al file video. Una scelta di questo tipo porta sia vantaggi che svantaggi:

Vantaggi:

- Le dimensioni del package rimangono contenute perché non devono essere fisicamente inclusi i files dei filmati.
- Tutti i riferimenti ad un particolare file video puntano ad una singola istanza del filmato situata su uno streaming server.

Svantaggi:

- Come già sottolineato, un riferimento ad una risorsa esterna all'interno di un package IMS o SCORM rende la fruibilità di tale pacchetto dipendente dalla disponibilità della risorsa esterna. Ad esempio, se lo streaming server su cui e pubblicato un filmato non è raggiungibile, tutti i corsi che hanno un riferimento al filmato non saranno pienamente fruibili.

Per poter creare un corso è necessario avere almeno il ruolo di Creators in modo da avere accesso all'interfaccia di gestione dei progetti. Un progetto è una classe che descrive un corso. Il corso prodotto eseguendo il build del progetto e un'istanza del progetto. Utilizzando lo stesso progetto, eventualmente modificandone alcune parti, è possibile produrre diverse istanze di corsi differenti. La modifica di un progetto non tocca le istanze di corsi già creati. Un modello simile è utilizzato per lo sviluppo di applicazioni software. Il progetto di corso sta al progetto di applicazione come il corso sta all'applicazione. L'operazione di build del corso può essere vista come la compilazione del codice del progetto che ha come output l'eseguibile dell'applicazione.
Si è scelto di utilizzare un modello di questo tipo per rendere totalmente indipendenti i corsi creati dai progetti che li hanno generati anche perché gli standard utilizzati richiedevano la creazione di un package che fosse il più indipendente possibile da entità esterne. In questo modo è inoltre possibile utilizzare lo stesso progetto come base di partenza per la creazione di un altro corso evitando di modificare i corsi precedentemente creati.

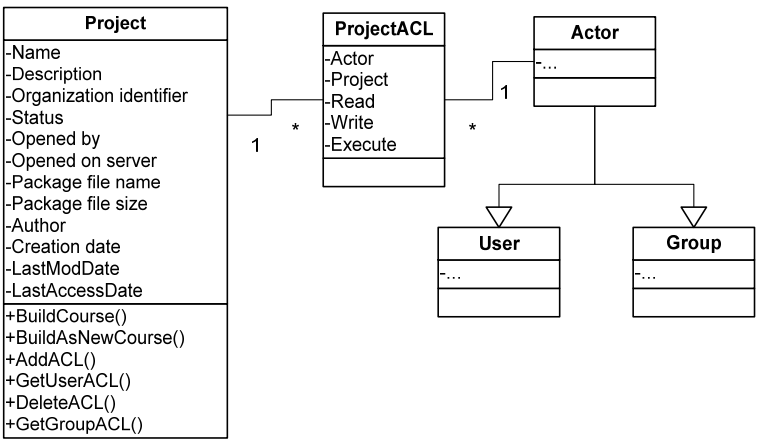


Figura 54: Diagramma della classe Project. I diritti vengono stabili mediante ACL.

Fisicamente un progetto è un package .zip registrato sul CM. La vita del progetto deve seguire una precisa serie di stati: alla sua apertura questo deve essere trasferito e decompresso dal CM sul server di front-end su cui si sta lavorando l'utente, l'editor dell'applicazione eseguirà le modifiche al progetto in locale. Quando il progetto sarà salvato verrà ricreato il package che conterrà le modifiche che sarà trasferito nuovamente sul CMS. La Figura 55 da una descrizione dettagliata degli stati un cui si può trovare un progetto e delle operazioni che sono eseguite dal sistema durante i passaggi di stato. Quando un utente apre un progetto pone un lock su quest'ultimo in modo che altri utenti, pur avendone diritto, non vi possano lavorare contemporaneamente. Alla chiusura del progetto il lock viene rilasciato.

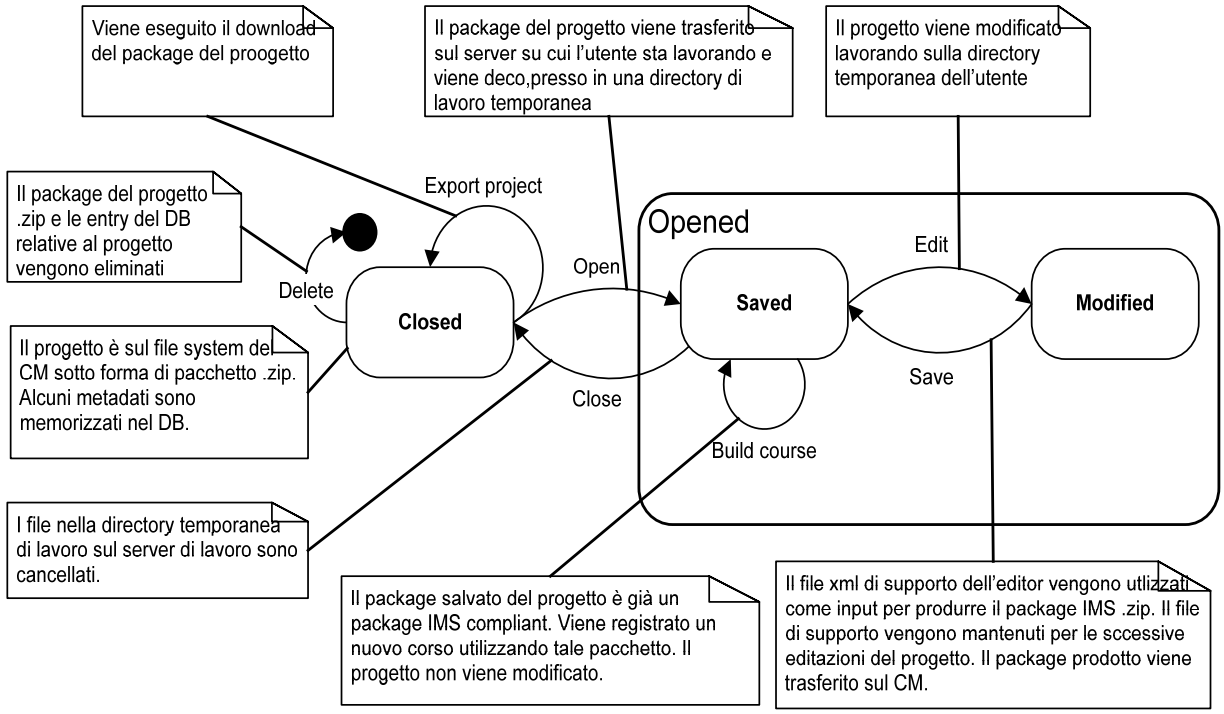


Figura 55: stati e transizioni di un progetto.

Quando il progetto è salvato tutte le modifiche vengono rese persistenti presso il CMS. Inoltre il formato interno usato dell'editor per la visualizzazione della struttura e dei contenuti del progetto in corso viene tradotto verso lo standard IMS-SCORM e i file delle risorse vengono pacchettizzate producendo un archivio .zip che contiene il progetto che, tuttavia, risulta gia essere un package IMS SCORM compliant.

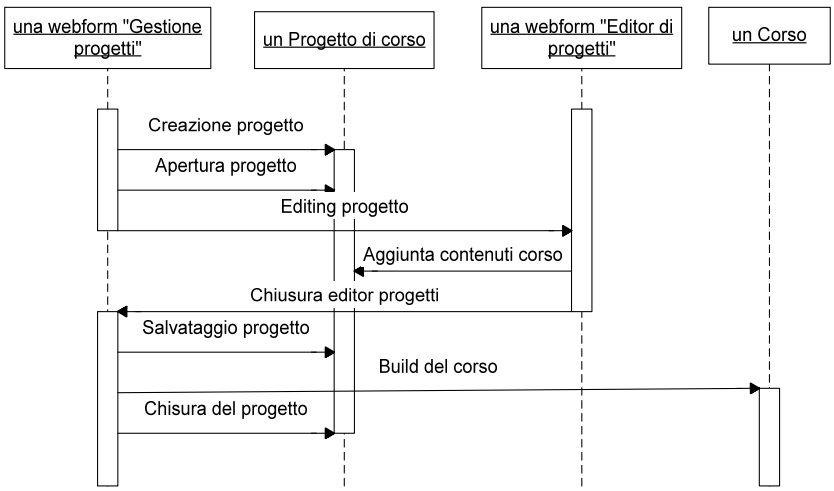


Figura 56: Creazione del progetto, editing e build del corso.

Il corso così creato sarà disponibile per il deployment. Con questa operazione il package del corso viene portato dal CMS sui server di front-end ospitanti l'ambiente di esecuzione lezi.NET Viewer e li decompresso. Il corso non risulta essere ancora pubblicato e quindi

fruibile dai normali utenti. Tuttavia il creatore potrà vederne e testarne un'anteprima nell'ambiente di esecuzione. A questo punto, eventualmente dopo aver distribuito i diritti sul corso mediante ACL, sarà possibile pubblicare il corso e renderlo fruibile a tutti gli utenti aventi diritti sufficienti.

La distribuzione dei diritti agli utenti mediante ACL, futuri fruitori di un corso, risulta essere un'operazione ripetitiva quando il numeri degli utenti è alto. Per ovviare a questo inconveniente è possibile creare un gruppo al quale si danno i diritti sul corso e poi metterci tutti gli utenti del corso. Lezi.NET, per aumentare ulteriormente il compito dei Creators, fornisce un'interfaccia che mediante un wizard permette dei gestire in modo guidato e automatizzato quest'ultima operazione. Gli utenti interessati ad un corso potranno effettuare una richiesta di sottoscrizione. Queste richieste, registrate dal sistema, saranno notificate al creatore del corso interessato. Quest'ultimo potrà, tramite il wizard, selezionare le richieste da soddisfare, creare un gruppo nuovo o utilizzarne uno preesistente, creare una nuova ACL da associare al corso e al gruppo.

Il ciclo di vita della fase di creazione di un corso è riassunto in Figura 57.

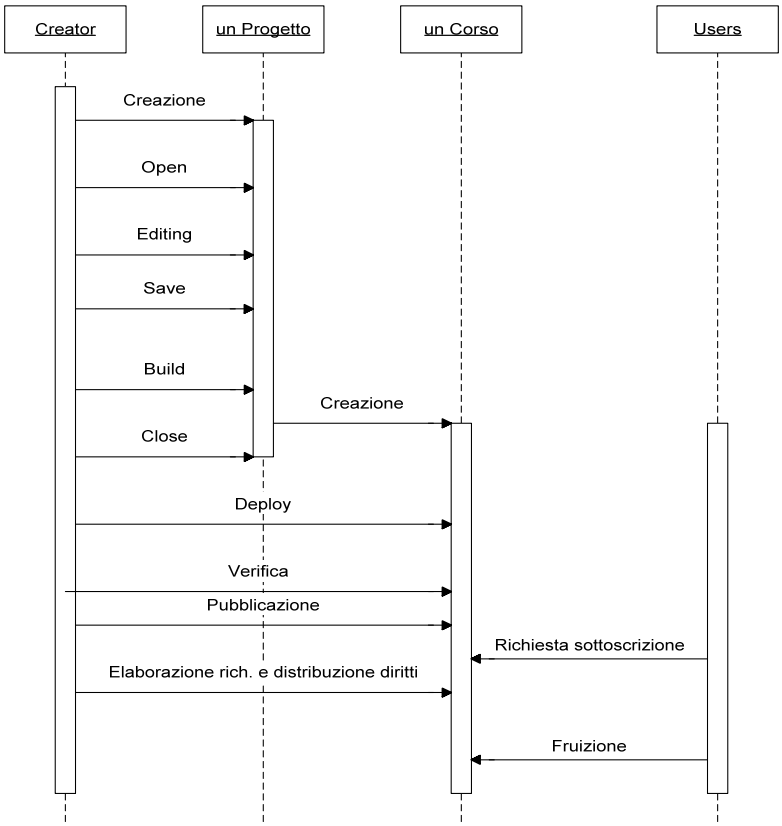


Figura 57: Creazione e pubblicazione di un corso.

Il sistema mette a disposizione del creatore due strumenti: un sistema di news da pubblicare per notificare agli utenti, ad esempio, del sistema la pubblicazione di un nuovo corso ed un sistema per registrare eventuali commenti riguardanti i corsi pubblicati.

8.1.4 Lezi.NET Administrator

Lezi.NET administrator è la parte di applicazione Web che permette all'amministratore di sistema e all'utente creatore di gestire gli account utente, gruppi, richieste di sottoscrizione e distribuzione di diritti utente. Sono principalmente delle pagine aspx che permettono mediante svariate forms e wizard di inserire dai nell'applicazione.

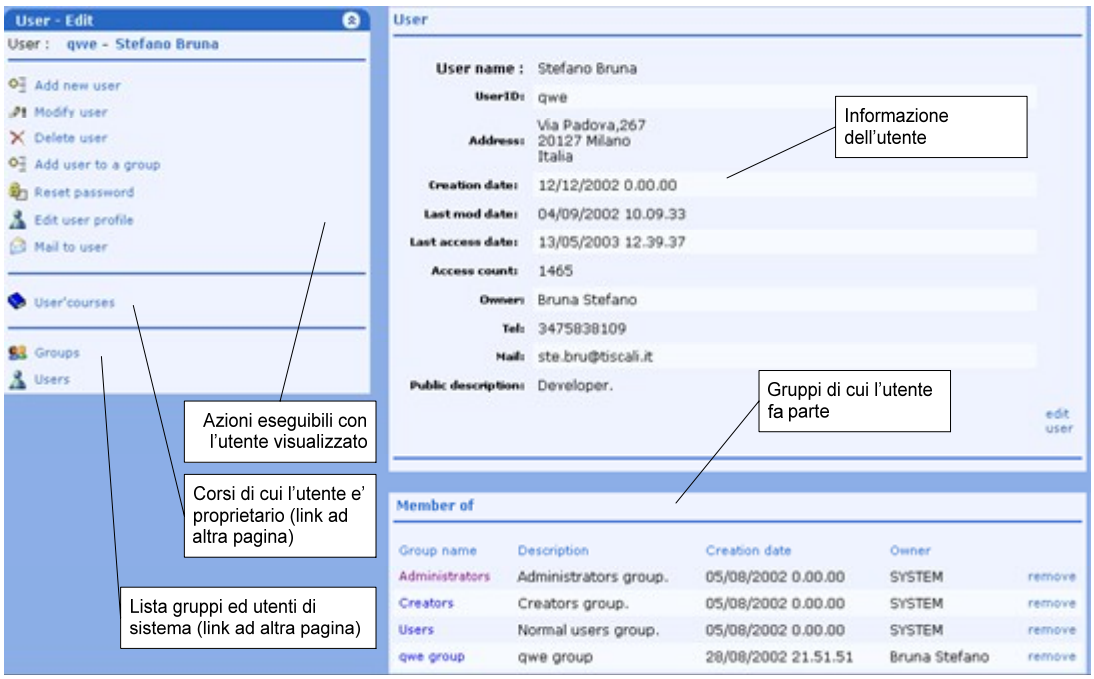


Figura 58: Pagina per la gestione dell'utente. Ad tale pagina si arriva selezionando un utente nella pagina che elenca gli utenti presenti nel sistema.

Tutti i dati inseriti vengono sottoposti ad una prima validazione che controlla la presenza ed il formato dell'input utente. Per effettuare queste operazioni il. Framework .NET fornisce due componenti molto utili ed importanti. Il requiredDataValidator ed il dataFormatValidator. Il primo verifica semplicemente la presenza di un dato considerato obbligatorio in una form. Tuttavia il componente risulta essere utile in quanto è in grado di visualizzare automaticamente un messaggio di errore quando l'utente trascura un campo di una form. Il controllo e la visualizzazione del messaggio può avvenire client side (mediante codice Jscript generato automaticamente). Il secondo componente, il dataFormatValidator , opera invece server-side e, mediante l'utilizzo di espressioni regolari, è in grado di verificare il formato dei dati. Questa operazione, spesso trascurata, risulta essere molto importante da punto di vista della sicurezza. Come espresso in Writing secure code (Vedi SECURE) "All input is evil". Durante il post di dati provenienti da form avvengono infatti attacchi come la SQL injection o il buffer overrun. Durante una SQL injection l'utente che 'attacca' il sito scrivendo in un campo di una form un stringa che mediante l'opportuno uso di apici (") permette di modificare la query string. Ad esempio, in una pagina di login, la query string per la verifica della password potrebbe essere:

```
String queryString = "SELECT * FROM Utenti WHERE User = " userid.toString() + "
AND Pswd = " + pass.toString();
```

Verificando che il risultato della query sia un recordset non vuoto si permette l'accesso all'utente. Tuttavia inserendo la seguente stringa al posto della password:

```
miaPass  + " OR User = 'MioUser'"
```

si è sicuri che almeno un record venga ritornato garantendosi cosí l'accesso al sito.

Il dataFormValidator,verificando la stringa inserita, permette di evitare attacchi di questo tipo. In lezi.NET esiste tuttavia un secondo livello di validazione ottenuto 'gratuitamente' grazie all'utilizzo delle Stored procedure che controllano il tipo ed i formato dei parametri prima dell'esecuzione della query.

Attacchi come il buffer overrun, se vengono correttamente gestite le eccezioni, grazie all'utilizzo di un linguaggio come il C# e al run-time che ne esegue l'IL (ByteCode) risultano essere inefficaci.

Una volta validati, i dati inseriti vengono utilizzati per la costruzione di oggetti che sono poi passati, mediante Web services, al lezi.NETCMS. Tutta la business logic che esegue verifiche di autenticazione, autorizzazione e consistenza è presente, come sarà raccontato in seguito, a questo livello.

Esempio 18. Durante la modifica dei dati di un utente la form viene compilata con i dati correnti. Dopo la modifica da parte dell'utente i dati vengono rispediti al CMS.

```
leziUser user =
Global.cmsvcAPI.GetUser(Convert.ToInt32(this.Request.QueryString["IDUser"]),(int)
Session["IDUser"],Session["token"].ToString()); // ottengo l'utente dall'API del
CMS

// compilo la form
```

Al successivo post back dopo il commit dei dati…

```
try          {
             Global.cmsvcAPI.UpdateUser(this.newEditUserControl.User,(int)Session
["IDUser"],Session["token"].ToString());
             this.Response.Redirect("User.aspx?IDUser=" +
this.Request.QueryString["IDUser"].ToString());
             }
             catch (SoapException cme)
             {
             Global.LogExceptionOnFile("leziNET.ProfileForm:UserStat",cme);
             if (Global.Debug) throw cme;
             else
                    {
                           this.DisplayErrorMessage(cme.Message);
                           return;
                    }
             }
```

Come si può notare dall'esempio X, per le operazioni con il CMS, è necessario passare come argomenti IDUser e token per poter eseguire sul CMS l'identificazione dell'utente e quindi l'autorizzazione per le operazione.

8.1.5 Lezi.NET streaming server

L'utilizzo di uno streaming server si è reso necessario per rendere efficiente la fruizione di contenuti il cui media principale è un filmato o una traccia audio di lunga durata e quindi di grandi dimensioni. E' infatti impensabile che una volta nell'ambiente di esecuzione l'utente

sia costretto ad aspettare il download completo del filmato per poi iniziare l'esecuzione del contenuto. Alcuni player riescono tuttavia a simulare una sorta di streaming iniziando la riproduzione del filmato pochi instanti dopo l'inizio del download. Una soluzione di questo tipo ha però lo svantaggio di caricare il Web server che ad un certo punto si troverà ad dover gestire n flussi di dati per lunghi periodi di tempo, dovendo contemporaneamente servire le pagine Web richieste. Una soluzione che utilizza lo streaming server dedicato permette quindi di distribuire in modo intelligente il carico sia relativo alla banda che relativo alla potenza di elaborazione. Inoltre il player e lo streaming server si scambiano dei metadati per definire parametri, quali banda disponibile, durata della fase di buffering, qualità del canale di comunicazione, protocollo ottimale (UDP e TCP) etc, che permettono di migliorare l'esperienza dell'utente. Ad esempio, a causa di un momentaneo calo della disponibilità di banda, il player può decidere di diminuire il numero di frames visualizzati oppure di eseguire solo la traccia audio, in attesa che la banda necessaria per una normale esecuzione sia ripristinata.
Naturalmente lezi.NET non implementa un nuovo streaming server ma costruisce un wrapper per poter integrare uno streaming server esistente, Windows Media Service, nel sistema. Vedi Figura 59.

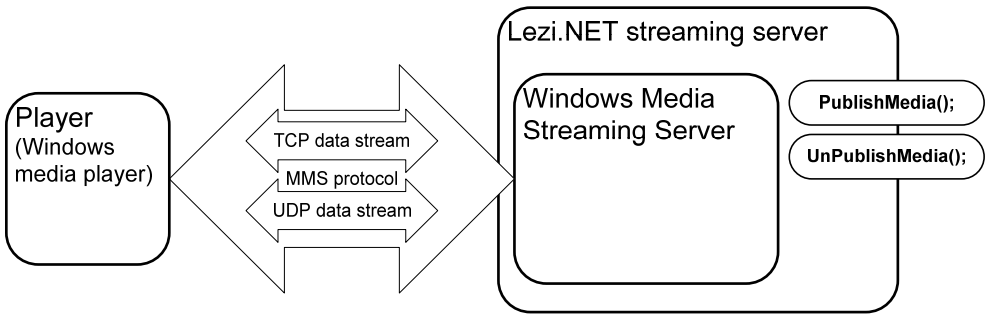


Figura 59: il contenitore implementato permette di integrare lo streaming server esistente nel sistema lezi.NET fornendo semplicemente una interfaccia per la pubblicazione dei contenuti.

L'utilizzo di Windows Media Streaming Server comporta alcune restrizioni relative al formato dei media distribuiti.
Lo streaming server, nell'architettura generale di un LMS, fa parte del delivery service in quanto anche'esso contribuisce alla distribuzione dei contenuti. Quindi il delivery service di lezi.NET è composto sia da un Web server per i media di tipo testuale e di uno streaming server par media quali audio e video.
Per ogni front tier è disponibile uno streaming server. Quando si esegue l'upload di un media presso un particolare front tier il media sarà fisicamente memorizzato presso lo streaming server corrispondente. Ciò non limita la scelta durante la fase di pubblicazione di un corso. Infatti quest'ultimo potrà essere pubblicato anche presso i front tier il cui streaming server non dispone dei media del corso poiché i riferimenti ai media originali rimangono costanti. Il corso può essere distribuito da un server lezi.NET mentre i filmati possono essere distribuiti da un altro server lezi.NET.

8.1.6 Proxy

Per implementare la comunicazione fra il front tier e il middle tier sono stati utilizzati i Web services. Gli estremi di questo canale di comunicazione sono i servizi del CMS di cui si parlerà in dettaglio nella Sezione 8.2 e l'ennesimo front tier, che costruendo messaggi XML-SOAP, è in grado di eseguire delle chiamate a metodi remoti. Per poter raccogliere tutta la logica necessaria per queste chiamate e nascondere la complessità al resto del front

tier si è creato un modulo software, una libreria a collegamento dinamico (dll), che funge da proxy per i servizi forniti dal CMS. Così facendo il front tier è convinto che i servizi che chiama siano locali poiché le chiamate che esegue sono chiamate ai metodi di un oggetto istanziato localmente e globalmente all'interno dell'applicazione.
Per poter costruire un oggetto di questo tipo in grado di "mascherare" servizi forniti tramite Web services esistono delle utilità fornite dal framework di sviluppo .NET.
Innanzitutto è necessario capire quali sono i servizi pubblicati. Per fare questo è stato usato un tool, Disco.exe, che permette di creare un documento che descrive i Web services pubblicati presso un url interrogato dall'utilità stessa. Questo documento descrittivo può essere utilizzato come input per un altro tool, wsdl.exe, che permette di creare codice sorgente che implementa un client per i servizi descritti. In pratica, crea un'insieme di classi, una per ogni tipo di oggetto utilizzato per la comunicazione dai servizi ed una classe che fornisce i servizi sotto forma di metodi. Compilando infine queste classi come libreria si ottiene il proxy desiderato. In lezi.NET ho utilizzato questo modo di procedere modificando il sorgente creato da wsdl.exe per poter rendere dinamico l'url(tramite metodo setServiceURL(…))a cui i servizi sono pubblicati. Non procedendo in questo modo sarebbe necessario ricompilare il proxy ogni volta che l'indirizzo della macchina che pubblica i servizi viene modificato.
Ho inoltre seguito la scelta di avvolgere il proxy in un successivo contenitore per poter astrarre ulteriormente i servizi offerti. Tale contenitore si occupa della gestione delle eccezioni di tipo remoto (SOAPException, TimeOutExcepiton etc) e di altri aspetti relativi alla comunicazione nascondendo definitivamente al resto dell'applicazione gli aspetti relativi ai Web services e alle chiamate remote. Tale modulo ha inoltre il pregio di permettere di fornire apparentemente più servizi dei Web service che nasconde. E', infatti, possibile realizzare un metodo che fornisce un servizio dato dal risultato di successive chiamate a diversi servizi. Vedi Figura 60.

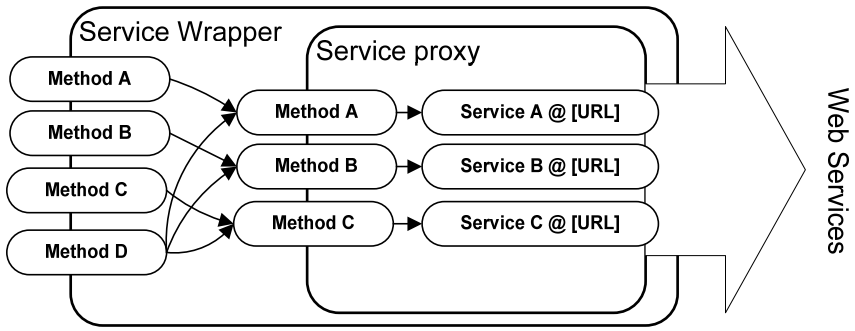


Figura 60: il proxy mappa uno a uno i servizi offerti su metodi di una classe.Il service wrapper utilizzando business logic e chiamate successive ai metodi A, B, C è in grado di fornire il servizio D.

8.2 Applicazione Web Lezi.NET content management system

L'applicazione lezi.NET Content Management System fornisce il supporto al front tier. I servizi, di cui si è parlato nella Sezione 7.2 , sono implementati mediante Web services pubblicati dal Web server IIS. Il CMS ha la struttura rappresentata in Figura 61. E' stato suddiviso il tre layer aventi funzionalità diverse. Il primo layer si occupa di fornire l'interfaccia per i diversi servizi. Ad esempio, il componente UsersAPi di Figura 61 fornisce un insieme di Web services che permettono al front tier di avere la possibilità di effettuare tutte le operazioni possibili con gli utenti.

Esempio 19. Un Web service del modulo API users.

```
[WebMethod]
public leziUser[] GetUsersList(int IDRole,UserFilterType filter,int
IDRoleReq,string token)
                    {
                           if (Core.Authentication.verifyToken(IDRoleReq,token) {
                                  if
((Core.Authorization.GetRightsUserToUsers(IDRoleReq) &
AuthorizationManager.RightsRead) == AuthorizationManager.RightsRead)
                                  {
                                         ArrayList al = null;
                                         if (filter.Code ==
UserFilterType.AllUsers.Code)
                                         {
                                                al = Core.Users.GetUsers();
                                         } else
                                         {
                                                al = Core.Users.GetUsers(IDRole);
                                         }
                                         leziUser[] lca = new leziUser[al.Count];
                                         for (int i=0; i < al.Count; i++)
                                         {
                                                lca[i] = (leziUser)al[i];
                                         }
                                         return lca;
                                  }
                                  else {
                                         throw new
UnauthorizedAccessException("Unauthorized access.");
                                  }
                           } else
                           {
                                  throw new UnauthorizedAccessException("Invalid
token.");
                           }
                    }
```

Come si può vedere dal codice dell'Esempio 19, lo usersAPI ha a disposizione gli oggetti Core.Users, Core.Authorization, Core.Authentication che permettono di accedere agli oggetti e ai metodi del layer di business logic che è il secondo livello del CMS. In questo layer è raccolta tutta la logica applicativa in modo da rendere il tutto il più modulare possibile. In questo modo una modifica all'interfaccia o al DBMS non comporta un cambiamento di questa parte. I due componenti più importanti del layer di BL sono il modulo di autenticazione e il modulo di autorizzazione.

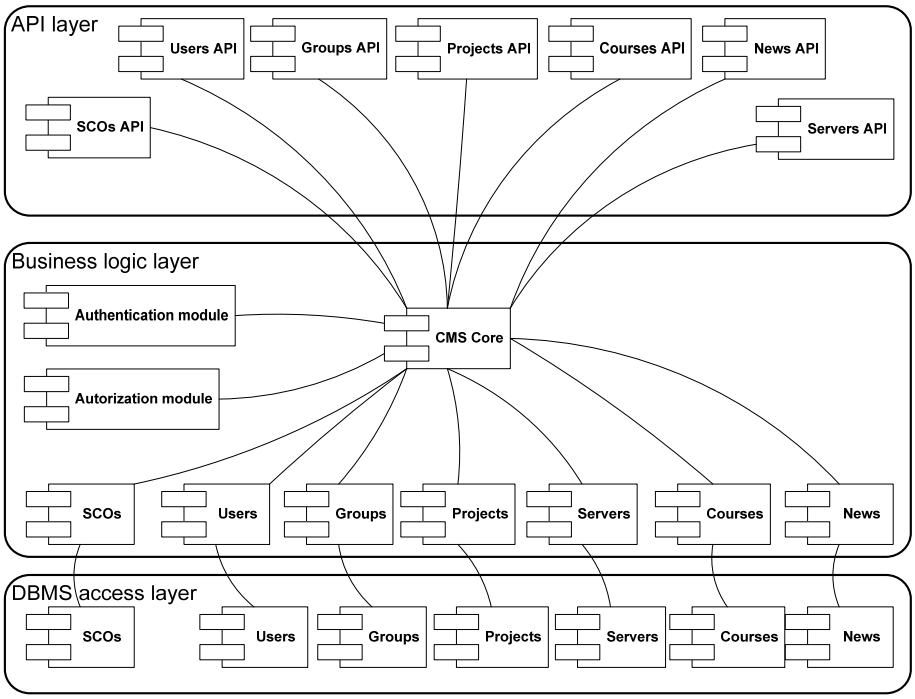


Figura 61: Diagramma di deployment dei diversi componenti di lezi.NETCMS.

8.2.1 Modulo di autenticazione

Primo compito di questo modulo è quello di gestire la fase di autenticazione dell'utente. I diversi front end rigirano la richiesta di login da parte di un utente al CMS. La richiesta di autenticazione è inviata spedendo fra i parametri la userID dell'utente e un hash della password. Il CMS cerca la password dell'utente con userID specificata, esegue lo stesso tipo di hash ed effettua un confronto. Se tale confronto ha esito positivo autentica l'utente ritornando un token che il front end dovrà utilizzare per poter avere accesso a tutti gli altri servizi offerti dal CMS.

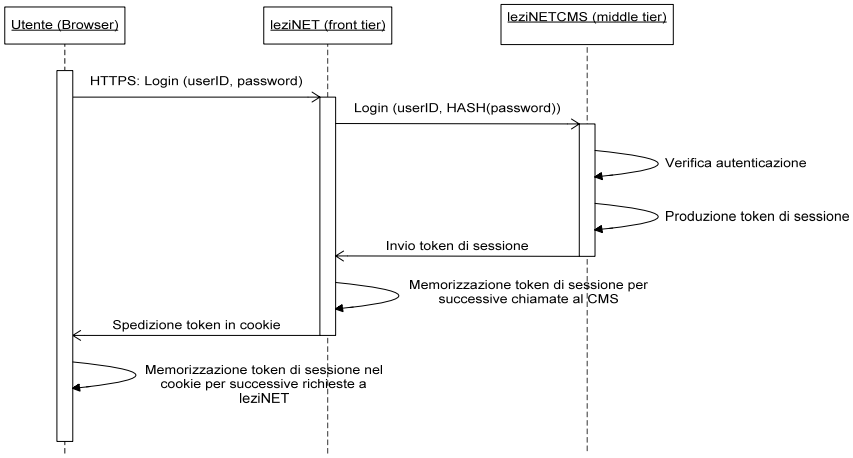


Figura 62: Sequence diagram della fase di autenticazione di un utente nel sistema.

Il token utilizzato dal front end per poter avere accesso ai servizi del CMS viene utilizzato come token anche dal sistema di autenticazione forms (vedi Sezione 8.1) che lo memorizza in un cookie sul client. In ogni Request del browser viene spedito il anche il token in modo da poter identificare univocamente l'utente che ha effettuato la richiesta.
Una volta effettuata l'autenticazione il modulo viene interpellato sempre per verificare che il token passato nei parametri di ogni richiesta corrisponda a quello che è stato fornito in precedenza dal sistema in modo da riuscire ad identificare l'utente senza che la password sia continuamente spedita diminuendo in questo modo i rischi di sniffing.

8.2.2 Modulo di autorizzazione

Dopo che una richiesta ad un servizio del CMS è stata "filtrata" dal modulo di autenticazione viene elaborata dal modulo di autorizzazione che in base al tipo di richiesta e al ruolo dell'utente decide se è possibile eseguire l'operazione. Per capire quali sono effettivamente i diritti di un utente su un oggetti è necessario eseguire l'algoritmo ricorsivo di Figura 20 nella Sezione 6.2. Una volta determinato il diritto dell'utente ad eseguire una determinata operazione la richiesta viene passata al layer che contiene i moduli contenenti la business logic.
Questi sistema permette di avere un ulteriore grado di sicurezza in quanto, anche se l'interfaccia grafica è in grado nascondere o disabilitare operazioni non eseguibili per un determinato utente, la possibilità di eseguire un operazione è verificata centralmente effettuando oltre che l'autorizzazione anche l'autenticazione. In questo modo si è sicuri che, anche se con un particolare serie di passaggi un utente "maligno" riuscisse ad avere un comando abilitato e visibile quando non dovrebbe esserlo, il CMS sarebbe in grado di bloccare l'operazione visualizzando un messaggio di errore ed eventualmente registrando in un log file il tentativo di "attacco".
Le access control list non vengono tuttavia utilizzate per tutti gli oggetti di sistema. Ad esempio, il diritto di poter inserire nuovi utenti in un sistema operativo e dato solo agli utenti appartenenti al gruppo Administrators (o root). Un sistema i cui accessi a tutti i suoi oggetti siano gestiti solo da ACL risulterebbe difficilmente gestibile. E' necessario quindi affiancare alle ACL anche le policy.

Il modulo di autorizzazione usa una lista di policies per decidere quali diritti attribuire ad un utente. Ad esempio esiste un policy che dice che gli utenti aventi ruolo di Administrators o Creators possono aggiungere utenti al sistema. Per informazioni relative alle policies si veda la sezione 6.2.

8.2.3 Moduli di gestione delle entità

Gli altri moduli del secondo layer contengono tutta la logica per la gestione dei diversi oggetti e le operazione eseguibili con tali oggetti. Ad esempio, il modulo che gestisce l'entità Corso deve provvedere alle funzionalità per l'importazione dei package SCORM. Infatti effettuato l'upload del package sul front-end, è necessario eseguire un trasferimento del file .zip sul middle tier. Una volta arrivato, viene estratto il manifest.xml e ne viene eseguito il parsing per leggere metadati utili all'operazione di archiviazione. Il manifest viene memorizzato in un campo del record relativo al corso che si sta importando. Se il contenuto del manifest è valido e il files che indicizza esistono realmente nel package del corso, il package stesso è memorizzato si file system con nome codificato (tipo CoursePack_243.zip) e viene creato il relativo record nel database. Inoltre, se si tratta di un corso SCORM compliant, il modulo per la gestione degli SCO di preoccupa, analizzando il manifest, di registrare gli SCO del corso inserendo per ognuno di essi un record nel database. Questo tipo di operazioni, eseguite nel layer di BL, sono supportate dal terzo layer che fornisce i metodi per l'accesso alla base di dati. Il terzo livello per poter accedere a sua volta al DBMS utilizza esclusivamente stored procedure. Questo è il livello che, in caso di cambiamento del DBMS deve essere sostituito. Infatti, non utilizzando query SQL standards si è legati fortemente al tipo di data base scelto durante lo sviluppo poiché proprio nel codice di questo livello si preparano, tramite le API ADO.NET, le chiamate alle stored procedure. Per maggiori dettagli consultare la Sezione 8.3.2

8.3 Base di dati

Il terzo tier dell'applicazione è composto dalla base di dati che memorizza lo stato del sistema. Parte dei degli schemi presenti in questa sezione appartiene il realtà alla parte relativa alla progettazione. Tali schemi sono stati spostati qui per alleggerire le sezioni precedenti e poter sfruttare la capacità descrittiva dei documenti di progetto per analizzare e rappresentare l'implementazione realizzata.
I dati contenuti nel database non vengono visti come recordset. Durante, ad esempio, la lettura di dati mediante stored procedure, i record trovati vengono immediatamente utilizzati per instanziare una classe o una collezione di classi che rappresentino, in modo strutturato, i dati ottenuti. Questo passaggio ha un piccolo impatto sulle prestazioni ma tutto il resto del sistema beneficia dei vantaggi della programmazione OO. Il codice scritto risulta più chiaro e quindi facile da comprendere, modificare e manutenere.

8.3.1 Schema

Nella Figura 63 è presentato lo schema della base di dati utilizzata. E' stata utilizzata una notazione sintetica per permettere l'impaginazione. La consultazione dello schema è abbastanza intuitiva. La sigla PK sta per Primari Key e la sigla FK sta per Foreign key. E' possibile comprendere completamente le relazioni aiutandosi con i nomi delle tabelle e dei campi.
Come già spiegato nella Sezione 6.4 la tabella Entities permette di avere un unico identificatore sia per Users che per Groups. In questo modo è facile implementare

l’annidamento di gruppi utilizzando la tabella EntityMembership in cui nella prima colonna è specificato l’identificatore di un gruppo mentre nella seconda colonna potrà essere presente l’identificatore di un gruppo oppure di un utente.
La tabella EntityACL avrebbe dovuto servire per regolare i diritti di modifica sugli utenti da parte di altri utenti. Per non rendere eccessivamente complessa la gestione del sistema si è optato invece per l’utilizzo di una policy che permetta agli utenti con il ruolo di Administrators di modificare un qualsiasi utente e agli utenti con il ruolo di Creator di modificare solo gli utenti di cui è il possessore.
Altra tabella fondamentale è Courses che insieme a CoursesSystem raccoglie le informazioni dei corsi presenti nel sistema. In CoursesSystem si può vedere il campo che memorizza il percorso di file system del package IMS/SCORM del corso. Anche la tabella Project ha una tabella in cui sono memorizzati dati di sistema chiamata ProjectSystem. Le tabelle Users_SCO, SCOs e Courses permettono tramite le loro relazioni di implementare la registrazione degli SCO per i corsi SCORM e la memorizzazione della struttura xml CMI per la memorizzazione dello stato di esecuzione di uno SCO da parte di un utente. Si può vedere infatti nella tabella Users_SCO il campo CMIXml in cui è memorizzato il documento xml che modella la struttura CMI (Vedi Sezione 6.4.1).
Le tabelle che terminano con la dicitura ACL mettono in relazione un oggetto con un gruppo o con un utente specificando nella tabella, mediante tre boolean il tipo di accesso.

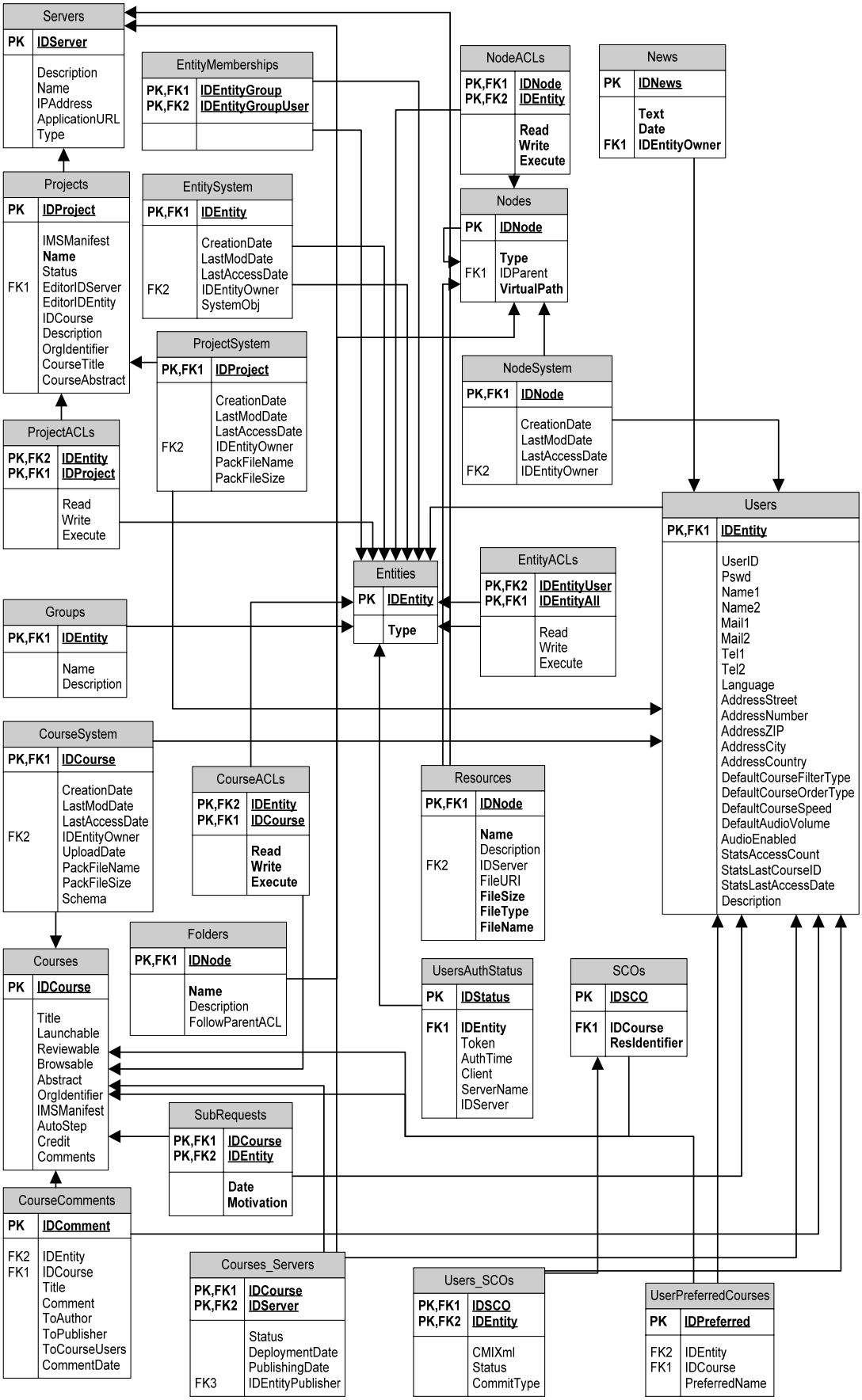


Figura 63: Schema del database.

8.3.2 Stored procedure

Per accedere alla base di dati il middle tier, leziNETCMS, utilizza esclusivamente stored procedure. I motivi di tale scelta sono stati:

- Desiderio di sperimentare le nuove classi ADO.NET per l'accesso al DB mediante stored procedure.
- Possibilità di valutare pro e contro di tale scelta. L'utilizzo delle query string, come termine di paragone, l'avevo già sperimentata in altri progetti.
- Buone performance garantite dal fatto che il codice delle stored procedure risiede ed è già pre-compilato presso il DB.

Tale scelta ha, comunque, anche lo svantaggio di rendere l'applicazione meno portabile. In poche parole l'utilizzo delle stored procedure mediante ADO.NET e SQLServer rende il sistema dipendente dalla piattaforma Microsoft. Tuttavia l'architettura modulare di leziNETCMS (vedi Figura 61) ha lo scopo di minimizzare l'impatto di questa scelta. Volendo cambiare tipo di base di dati è necessario riscrivere solamente alcuni moduli. Nell'Esempio 20 è possibile vedere una stored procedure che permette di creare un corso.

Esempio 20: Stored procedure scritta in Transact-SQL per la creazione di un corso.

```
CREATE PROCEDURE SetCourse
@title nvarchar(100),
@orgIdentifier nvarchar(255),
@abstract nvarchar(255),
@comments nvarchar(255),
@schema nvarchar(50),
@packFileName nvarchar(255),
@packFileSize bigint,
@browsable bit,
@launchable bit,
@reviewable bit,
@credit bit,
@autoStep bit,
@IDRoleOwner int,
@manifest ntext,
@creationDate datetime,
@lastModDate datetime,
@lastAccessDate datetime,
@uploadDate datetime,

@id int OUTPUT

 AS
INSERT INTO Courses
                      (Title, Launchable, Reviewable, Browsable, Abstract,
OrgIdentifier, IMSManifest, Credit, AutoStep, Comments)
VALUES     (@title, @launchable, @reviewable, @browsable, @abstract,
@orgIdentifier, @manifest, @credit,@autoStep, @comments)

SELECT @id = @@IDENTITY


INSERT INTO CourseSystem
                      (IDCourse, CreationDate, LastModDate, LastAccessDate,
IDRoleOwner, UploadDate, [Schema], PackFileName, PackFileSize)
VALUES     (@id, @creationDate, @lastModDate, @lastAccessDate, @IDRoleOwner,
@uploadDate, @schema, @packFileName, @packFileSize)
GO
```

Nel precedente esempio si può notare come i parametri utilizzati delle istruzioni `Insert` preceduti dal carattere "@" siano dichiarati e il loro tipo sia specificato. Ciò fornisce alle stored procedure, e quindi a tutto il DB un'interfaccia chiara e non scavalcabile. Questa soluzione permette di facilitare moltissimo il compito dello sviluppatore che non si trova ad avere i tipici problemi riguardanti le conversioni di tipo fra i dati nel DB e gli oggetti dell'applicazione.
Le librerie ADO.NET, inoltre, forniscono classi specifiche per l'utilizzo delle stored procedure e mediante appositi metodi permettono di risolvere il problema delle conversioni. Vedi Esempio 21.

Esempio 21: Codice C# per la creazione di un corso. Vengono utilizzate le librerie ADO.NET per l'accesso al DB tramite stored procedure. Le stesse librerie possono, comunque, essere utilizzate anche per l'accesso mediante query string.

```
private int SetCourse(leziCourse course,string manifestString){
              SqlConnection conn = new SqlConnection(m_connectionString);
              SqlCommand cmd = new SqlCommand("SetCourse",conn);
              cmd.CommandType = CommandType.StoredProcedure;
              SqlParameter pTitle = cmd.Parameters.Add("@title",course.title);
              pTitle.Direction = ParameterDirection.Input;

              SqlParameter pOrgID =
cmd.Parameters.Add("@orgIdentifier",SqlDbType.NVarChar);
              if (course.orgIdentifier == null) pOrgID.Value = System.DBNull.Value;
              else pOrgID.Value = course.orgIdentifier;
              pOrgID.Direction = ParameterDirection.Input;

              SqlParameter pAbstract =
cmd.Parameters.Add("@abstract",course.courseAbstract);
                           pAbstract.Direction = ParameterDirection.Input;

              SqlParameter pComments = cmd.Parameters.Add("@comments",SqlDbType.NVarChar);
              if (course.Comments == null) pComments.Value =  System.DBNull.Value;
              else pComments.Value = course.Comments;
              pComments.Direction = ParameterDirection.Input;

              SqlParameter pAutoStep = cmd.Parameters.Add("@autoStep",course.autoStep);
              pAutoStep.Direction = ParameterDirection.Input;

              SqlParameter pBrows = cmd.Parameters.Add("@browsable",course.isBrowsable);
              pBrows.Direction = ParameterDirection.Input;

              SqlParameter pLaunch = cmd.Parameters.Add("@launchable",course.isLaunchable);
              pLaunch.Direction = ParameterDirection.Input;

              SqlParameter pCredit = cmd.Parameters.Add("@credit",course.Credit);
              pCredit.Direction = ParameterDirection.Input;

              SqlParameter pReview = cmd.Parameters.Add("@reviewable",course.isReviewable);
              pReview.Direction = ParameterDirection.Input;

              SqlParameter pIDRoleOwner =
cmd.Parameters.Add("@IDRoleOwner",course.System.IDRoleOwner);
                           pIDRoleOwner.Direction = ParameterDirection.Input;

              SqlParameter pCDate = cmd.Parameters.Add("@creationDate",SqlDbType.DateTime);
              pCDate.Value = DateTime.Now;
              pCDate.Direction = ParameterDirection.Input;

              SqlParameter pLMDate = cmd.Parameters.Add("@lastModDate",SqlDbType.DateTime);
              pLMDate.Value = DateTime.Now;
              pLMDate.Direction = ParameterDirection.Input;

              SqlParameter pLADate =
cmd.Parameters.Add("@lastAccessDate",SqlDbType.DateTime);
              pLADate.Value = DateTime.Now;
              pLADate.Direction = ParameterDirection.Input;

              SqlParameter pUDate = cmd.Parameters.Add("@uploadDate",SqlDbType.DateTime);
              pUDate.Value = DateTime.Now;
              pUDate.Direction = ParameterDirection.Input;
```

```
                SqlParameter pCPFN =
cmd.Parameters.Add("@packFileName",course.System.PackFileName);
                                pCPFN.Direction = ParameterDirection.Input;

                SqlParameter pCPFS =
cmd.Parameters.Add("@packFileSize",course.System.PackFileSize);
                                pCPFS.Direction = ParameterDirection.Input;

                SqlParameter pSchema = cmd.Parameters.Add("@schema",course.System.Schema);
                                pSchema.Direction = ParameterDirection.Input;

                SqlParameter pIdentity = cmd.Parameters.Add("@id",SqlDbType.Int);
                                pIdentity.Direction = ParameterDirection.Output;

                SqlParameter pMan = cmd.Parameters.Add("@manifest",SqlDbType.NText);
                                pMan.Value = manifestString;
                                pMan.Direction = ParameterDirection.Input;

            try {
                                conn.Open();
                                cmd.ExecuteNonQuery();
                                return (int)pIdentity.Value;
                        }
                        catch (SqlException sqle)
                        {

                Global.LogExceptionOnFile("leziNETCMwsvc:CoursesManager:SetCourse","123",sqle
);
                throw;
                        }
                        catch (InvalidOperationException ioe) {
                Global.LogExceptionOnFile("leziNETCMwsvc:CoursesManager:SetCourse","123",ioe)
;
                throw;
                        }
                        catch (Exception e){
                Global.LogExceptionOnFile("leziNETCMwsvc:CoursesManager:SetCourse","123",e);
                throw;
                        }
                        finally {
                                conn.Close();
                        }
```

9 Prestazioni

Un importante aspetto dell'applicazione riguarda le prestazioni. In una applicazione dall'architettura semplice in cui le pagine attive, interrogando la base di dati, presentano i dati all'utente non si può certo pretendere di introdurre grandi ottimizzazioni nel codice scritto dallo sviluppatore. Per aumentare il throughput, inteso come numero di richieste soddisfatte al secondo, a parità di piattaforma software, non rimane che scalare il sistema e/o aumentare la potenza dei server che ospitano l'applicazione. Per le applicazioni con architettura non elementare come lezi.NET è stato necessario, anche durante la fase di sviluppo, capire quale impatto sulle prestazioni potesse avere un particolare scelta di progetto. Basti pensare al fatto che quando l'utente accede alla pagina che presenta le informazioni di un corso. Il sistema, nel suo complesso esegue le seguenti operazioni:

- La richiesta dal client arriva a uno dei Web server del front-end.
- Il Web server verifica il token di autenticazione e, se valido, inizia l'elaborazione della pagina.
- La pagina in fase di elaborazione chiede al leziCMS informazioni riguardanti il ruolo dell'utente sul corso per poter renderizzare correttamente i vari componenti della pagina. La pagina è infatti personalizzata in base al ruolo dell'utente.
- La pagina chiede al CMS, tramite Web service, informazioni riguardanti il corso.
- Il CMS chiede alla base di dati il relativo record.
- Il CMS con i dati del recordset ritornato costruisce un oggetto che verrà successivamente serializzato in xml per essere restituito al front-end come parametro di ritorno di un Web service.
- Il front-end deserializza il documento xml e ricostruisce l'oggetto corso. Tale oggetto è utilizzato per visualizzare le informazioni del corso all'interno della pagina.
- Per poter visualizzare la tabella dei contenuti del corso la pagina chiede al CMS il manifest.xml del corso.
- Il CMS che riceve tale richiesta, sempre tramite Web service, esegue una query nel database estraendo il documento manifest.xml da un campo binary nel quale era stato "blobbato". Il documento viene estratto, memorizzato in una stringa e poi caricato come documento xml e restituito al front-end come parametro di ritorno di un Web service.
- La pagina esegue il parsing del documento manifest e mediante una procedura ricorsiva esegue il rendering delle varie voci della tabella dei contenuti.
- Dopo altre operazioni d minore importanza, la pagina viene finalmente spedita al client.
- Il client visualizza la pagina.

L'insieme delle operazioni che il sistema esegue ad ogni richiesta della pagina di un corso è quindi abbastanza corposo. E' necessario durante un'analisi delle prestazioni tenere conto che il codice delle pagine ASPX (l'intermediate code di .NET simile al bytecode di Java) non viene interpretato tutte la volte ma è eseguita una compilazione verso codice nativo alla prima richiesta della pagina. Le richieste successive provocheranno l'esecuzione di codice già compilato ed ottimizzato con un guadagno sostanziale in termini di prestazioni.

Per pagine che presentano dati con aggiornamenti non frequenti o per le quali la sincronia con la base di dati non è fondamentale, è possibile impostare anche delle politiche di caching server side in modo da evitarne la rielaborazione.

Tenendo conto di questi diversi aspetti sono stati eseguiti dei test per misurare il carico del sistema quando sottoposto ad un certo numero richiesta al secondo. Sono qui presentati due diversi tipi di test:

- Tempo di elaborazione e numero di richieste soddisfatte al secondo per un singola pagina.
- Numero di richieste soddisfatte al secondo per una navigazione tipo di un utente.

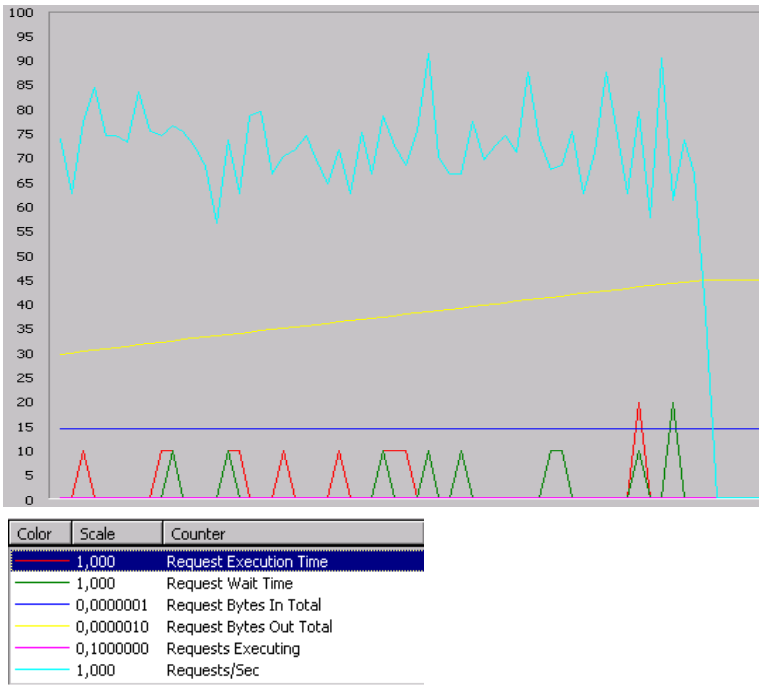
Per i due test di prestazioni è stato utilizzato lo stesso tool, il Microsoft Web stress tool e la piattaforma indicata nella tabella seguente.

Hardware:	
Processore	1 Intel Pentium 3 800 Mhz
Memoria	384 Mb
Software	
Sistema operativo	Windows 2000 Server SP3
DBMS	SQL Server 2000 SP3
Web server	IIS 5.0
Streaming server	Windows Media Server
.NET Framework	.NET Framework V1.0 SDK SP1

Il primo test ha i seguenti parametri per la configurazione che permettono di emulare una media di 75 richieste al secondo alla pagina Course.aspx:

Stress level (Threads)	6
Stress multiplier (Socket per thread)	6
Request delay	Random fra 10 e 300 ms
Pagina richiesta	Course.aspx per 4 volte.

Risultato:

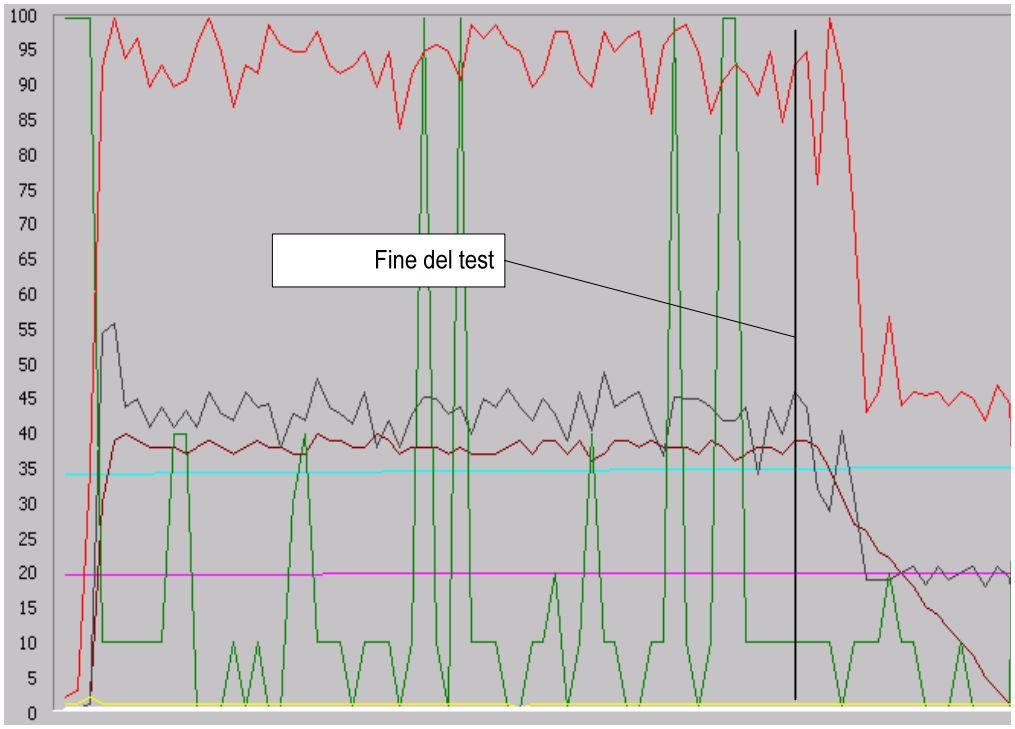


Come si può notare dal grafico nonostante arrivino 70/75 richieste al secondo (Requests/Sec) il tempo di esecuzione della singola risposta (Request Execution Time) non supera mai i 20 ms. Ciò è sottolineato inoltre dal fatto che il numero di richieste in esecuzione (Requests Executing) è sempre vicino a zero. Un altro indicatore non visualizzato nel grafico dimostra che i dati presentati durate il test non sono in una cache del Web server in quanto il numero di letture al database al secondo è abbastanza alto. Le performance sono buone poiché il codice delle pagina dopo essere stato compilato la prima volta viene riutilizzato per le richieste successive.

Il secondo test cerca di capire quale sia il collo di bottiglia del sistema eseguendo una serie di richieste a diverse pagine di corsi in sequenza.

Stress level (Threads)	20
Stress multiplier (Socket per thread)	10
Request delay	Random fra 10 e 300 ms
Pagina richiesta	Course.aspx per 6 volte su corsi sempre diversi.

Risultato:



Color	Scale	Counter
	1,000	% Processor Time
	1,000	Request Execution Time
	1,000	Request Wait Time
	1,000	Requests Queued
	0,0000001	Request Bytes In Total
	0,0000001	Request Bytes Out Total
	1,000	Requests Executing
	1,000	Requests/Sec

Pur arrivando un numero di richieste al secondo inferiori rispetto all'esempio precedente si nota come il tempo di esecuzione sia più alto causa della maggiore complessità del tipo di richiesta. Tuttavia il worker process .NET limita il numero di richieste accettate. Quelle che non sono elaborate sono accodate presso il Web server prima di essere inoltrate al motore d'esecuzione .NET. La dimostrazione di questo fatto è data dall'andamento della linea che indica le richieste in esecuzione (Request executing). Ad un certo punto, indicato dalla linea di fine test, nonostante il le richieste inoltrate dal tool di stress siano terminate, il Web server continua a soddisfare quelle in coda, facendo scendere rapidamente il numero di richieste in esecuzione fina a raggiungere lo zero.
Considerando anche altre prove, non incluse in questo documento, si può dedurre, guardando soprattutto come i diversi processi si spartiscono l'utilizzo del processore, che il collo di bottiglia è proprio il worker process .NET e quindi l'elaborazione della pagina. Il database e il Web server IIS influiscono solo per una percentuale al massimo del 20% - 30% nell'utilizzo del processore.

10 Piattaforma di sviluppo e piattaforma operativa

Attualmente il lato server del framework .NET è disponibile per piattaforma Windows. In particolare è disponibile nativamente su Windows 2003 Server e installabile su Windows 2000 Server. Non essendo, durante il periodo di sviluppo, ancora disponibile la versione 2003 del server di Microsoft la scelta Windows 2000 server è stata l'unica percorribile. Probabilmente in futuro, grazie al progetto open source Mono (Vedi MONO), potrebbe essere possibile installare l'applicazione anche su server Linux. Inoltre, data l'architettura di lezi.NET e la garanzia di interoperabilità data dai Web service è possibile utilizzare, per la stessa istanza di deployment, piattaforme differenti. Ad esempio, si potrebbero installare quattro server di front end su quattro server Linux e il CMS su piattaforma Windows 2000 Server. Un requisito piuttosto stringente riguarda invece il DBMS che deve essere un SQL Server 2000. Dato l'utilizzo di stored procedure scritte in T-SQL[36]. Un passaggio ad un altro DBMS causerebbe, infatti, la riscrittura delle classi di interfaccia del CMS con il database.

Piattaforme utilizzate:

Piattaforma A	
Hardware:	
Processore	1 Intel Pentium 4 1,7 Ghz
Memoria	384 Mb
Software	
Sistema operativo	Windows 2000 Server SP2
DBMS	SQL Server 2000 SP2
Web server	IIS 5.0
Streaming server	Windows Media Server
.NET Framework	.NET Framework V1.0 SDK

Piattaforma B	
Hardware:	
Processore	1 Intel Pentium 3 800 Mhz
Memoria	384 Mb
Software	
Sistema operativo	Windows 2000 Server SP3
DBMS	SQL Server 2000 SP3
Web server	IIS 5.0
Streaming server	Windows Media Server
.NET Framework	.NET Framework V1.0 SDK SP1

[36] *T-SQL. Transact SQL. Estensione di SQL.*

Piattaforma C	
Hardware:	
Processore	2 Intel Pentium 3 700 Mhz
Memoria	512 Mb
Software	
Sistema operativo	Windows 2000 Server SP3
DBMS	SQL Server 2000 SP3
Web server	IIS 5.0
Streaming server	Windows Media Server
.NET Framework	.NET Framework V1.0 SDK SP1
	.NET State server

Piattaforma D	
Hardware:	
Processore	1 Intel Pentium Xeon 2 450 Mhz
Memoria	256 Mb
Software	
Sistema operativo	Windows 2003 Server RC2
DBMS	SQL Server 2000 SP3
Web server	IIS 6.0
Streaming server	Windows Media Server
.NET Framework	.NET Framework V1.0 SDK SP2

Configurazioni di testing:

Configurazione	Tipo di test	Piattaforma
Configurazione 1	Alpha e Beta Test	Piattaforma A
Configurazione 2	Release Candidate Test	Piattaforma B
Configurazione 3	Test di deployment multiprocessore	Piattaforma C
Configurazione 4	Test su Windows 2003 Server e IIS 6	Piattaforma D
Configurazione 5	Test di deployment con distribuzione dei diversi servizi su diverse piattaforme	Piattaforma B per applicazioni, Piattaforma C per DBMS
Configurazione 6	Test di deployment con distribuzione dei diversi servizi su diverse piattaforme	Piattaforma B per applicazione front tier leziNET, Piattaforma C per applicazione middle tier leziNETCSM e DBMS.

Le configurazioni 1 e soprattutto la 2 sono state testate ampiamente. La 3,4,5,6 sono state essenzialmente prove di deployment per verificare la compatibilità con ambienti sostanzialmente diversi da quello in cui l'applicazione era stata sviluppata. Durante il testing in configurazione 2 sono state effettuate diverse correzioni di errori nel software evidenziate da malfunzionamenti venuti alla luce grazie all' utilizzo dell'applicazione da parte di diversi utenti. Il feedback riguardante la usability è stato positivo soprattutto tenendo conto del fatto che gli utenti utilizzatori, pur non avendo ancora a disposizione un manuale, riuscivano ad operare facilmente dopo un breve tutorial. Dopo alcuni mesi di utilizzo l'applicazione si è dimostrata stabile e non sono stati riscontrati ulteriori

malfunzionamenti. Tuttavia è da sottolineare il fatto che la fase di testing è stata pianificata tenendo conto del fatto che l'intero progetto è un progetto di ricerca e non un progetto per la produzione di un sistema commerciale. Infatti, rispetto al tempo di sviluppo, circa dieci mesi, le risorse dedicate alla fase di testing sono state limitate ed insufficienti. Per questo motivo il sistema é da considerare ancora una Release Candidate e non una Release to Manufacturer. L'utilizzo da parte di diversi utenti, le eventuali segnalazioni di malfunzionamenti e le conseguenti correzioni continuano tuttora.

Per una configurazione semplice di deployment si veda Figura 64

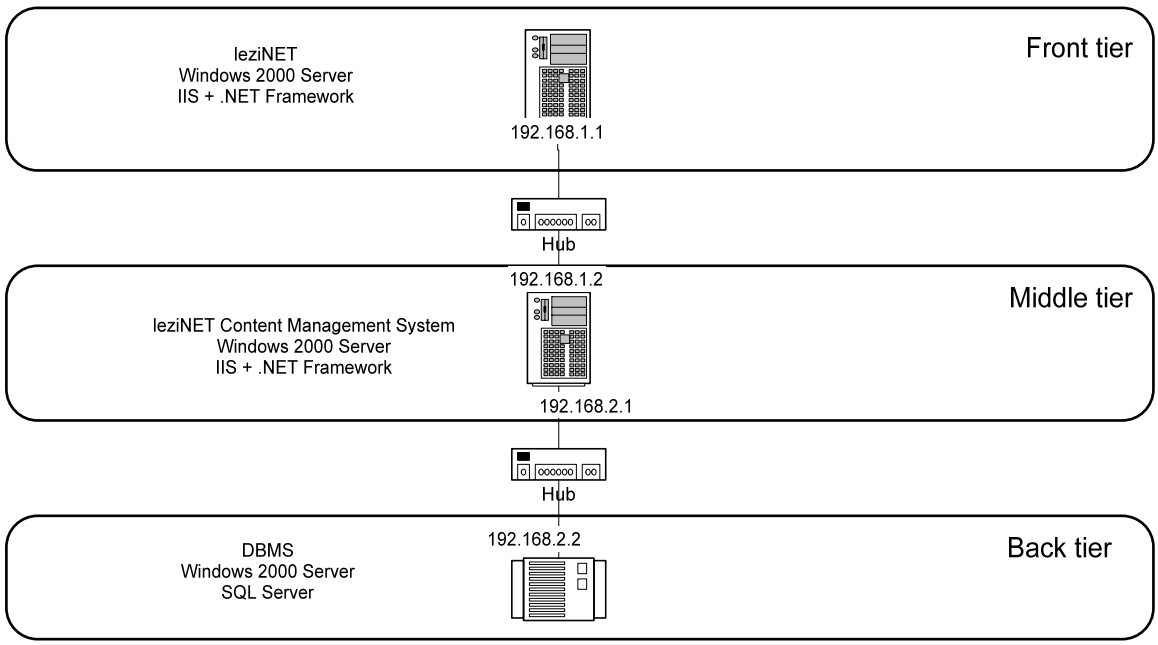


Figura 64: Semplice configurazione di deployment dell'LMS lezi.NET. Non é tuttavia la configurazione più elementare: è possibile avere tutti i tre tier su un singolo server.

lezi.NET si serve, durante l'elaborazione delle pagine, della variabile di sessione Session[37] che memorizza le informazioni della sessione per l'utente. Questa variabile dà la possibilità di mantenere lo stato durante la navigazione dell'utente fra le varie pagina pur utilizzando un protocollo di comunicazione stateless come l'HTML. In tale variabile è possibile, ad esempio, memorizzare un identificatore dell'utente in modo da conoscere l'identità dell'utente che ha eseguito la richiesta per una pagina aspx. L'utilizzo di tale variabile potrebbe risultare problematico quando si intende eseguire il deployment dell'applicazione su server multiprocessore. Per ogni processore del server IIS istanzia un worker process che si occuperà di elaborare le richieste dei client per le pagine .NET aspx (Vedi Figura 65).

[37] *La variabile é una collezione di oggetti di tipo Object. E' possibile quindi immagazzinare qualsiasi tipo di istanza di classe purché sia serializzabile.*

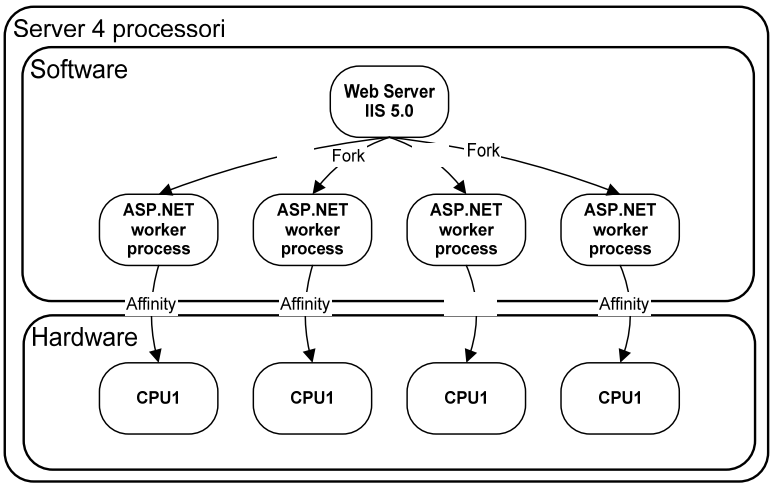


Figura 65: L'affinità di un worker process è una parametro largamente configurabile mediante CPUMASK bit che possono essere settate nel file di configurazione .NET di sistema machine.config. In IIS 6 invece i pool di worker process e le affinità con i vari processori possono essere settate dall'interfaccia di gestione dei servizi internet di sistema.

Per ogni worker process esiste una raccolta di variabili di sessione. Le diverse raccolte appartenenti a diversi worker process non condividono tuttavia i loro dati. Se una richiesta per una pagina venisse soddisfatta da un worker process e una successiva da un altro worker process i valori memorizzati nella variabile di sessione dal primo non sarebbero visti dal secondo con conseguente perdita di informazioni.
La configurazione di default per IIS+.NET non prevede l'utilizzo del Web gardening[38] e quindi un singolo worker process è in esecuzione sulla macchina anche se si tratta di un server multiprocessore. Quindi per poter rendere l'applicazione scalabile è necessario spostare la gestione della sessione dall'interno del worker process[39] a qualche servizio esterno che funzioni come storage comune per tutti i worker process in esecuzione sulla macchina.
Tale servizio è installato insieme al framework anche se di default è disabilitato. Il passaggio da una configurazione all'altra richiede solo la modifica di un parametro di configurazione dell'applicazione e l'avvio dell'ASP.NET State service. Questa configurazione ha il vantaggio di permettere ad un'applicazione che usa la variabile Session di essere largamente scalabile poiché è possibile sia il Web gardening che il clustering/ di Web server. (Vedi Figura 66) . Lezi.NET è stato installato e configurato anche su una macchina multiprocessore utilizzando il Web gardening e lo state server. Questa operazione è stata possibile senza dover modificare l'applicazione. Solo nel file di configurazione Web.config è stato modificato l'attributo **mode**.

```
<configuration>
  <sessionstate
      mode="stateserver"
      cookieless="false"
      timeout="20"
       server="127.0.0.1"
      port="42424"
  />
</configuration>
```

[38] *Server web multiprocessore.*
[39] *Questo tipo di gestione della variabile di sessione è detta "InProc".*

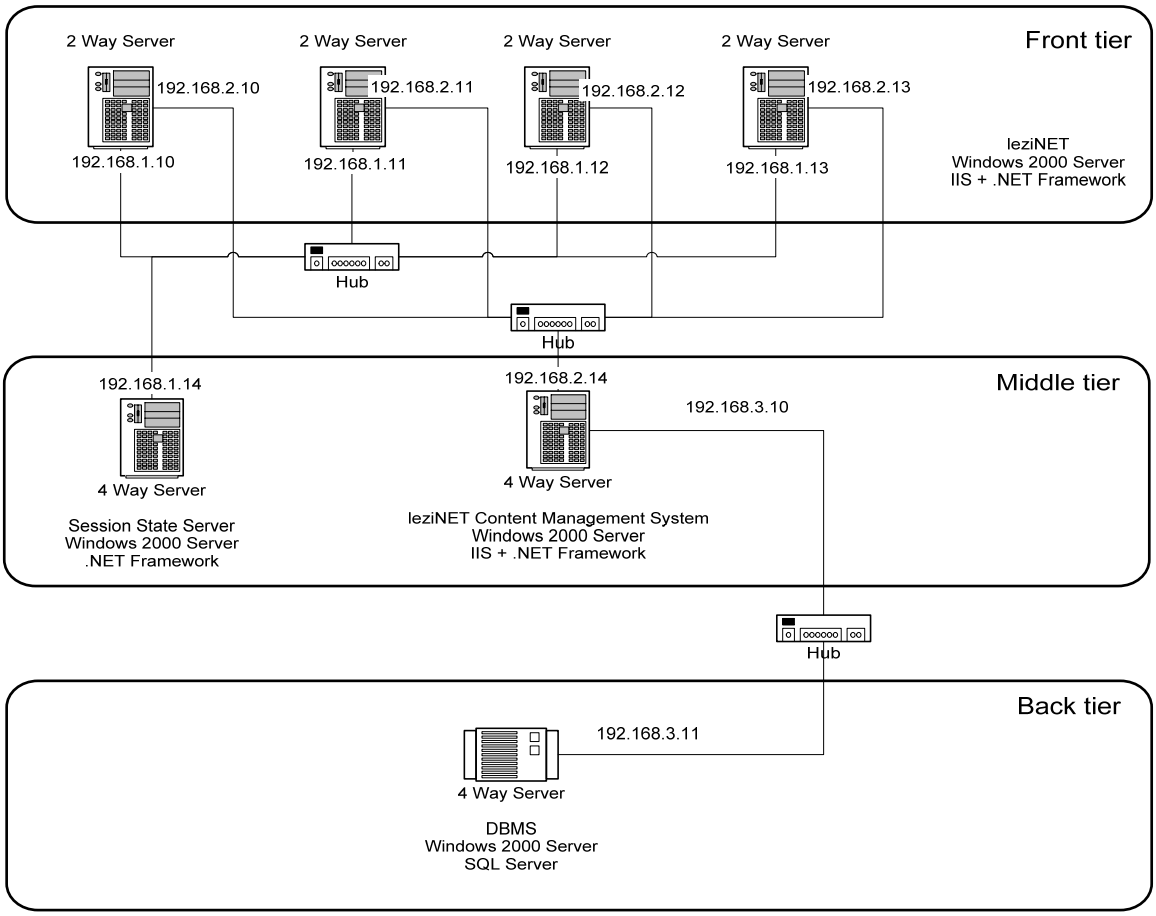


Figura 66: Questa configurazione prevede l'utilizzo dello state server per la gestione della sessione. La gestione del clustering dei server del front tier é gestita da leziNETCMS.

L'utilizzo del session state server non è l'unica alternativa per le gestione delle variabili di sessione degli utenti. E' possibile utilizzare anche un DBMS per poter memorizzare passo dopo passo l'evolversi dello stato delle sessioni. Una configurazione di questo tipo, tuttavia, sembra essere più adatta a sistemi per i quali la persistenza dei dati di sessione è fondamentale. Un LMS non ha sicuramente un requisito di questo tipo poiché la perdita delle informazioni di sessione non causa danni irreparabili. Per l'utente si tratta, infatti, di rieffettuare la login e riprendere il lavoro da dove si era lasciato. Per questo motivo lezi.NET non è stato testato in configurazione "SQLServer".
Un'ultima nota riguarda le prestazioni delle tre alternative. La soluzione più performante risulta essere quella inProc in cui la collezione delle sessione è memorizzata nello stesso processo del worker process. Passando allo state server si ha una calo delle prestazioni, a parità di configurazione, del 15%. Sempre a parità di configurazione la soluzione con il DBMS porta ad un calo del 20%. Tuttavia le due ultime soluzioni permettono una grande scalabilità e quindi sono le soluzioni preferibili per sistemi di grandi dimensioni richiedenti una certa qualità di servizio. Per i dettagli relativi al deployment di tutta l'applicazione si rimanda all'Appendice C.

11 Validazione e sperimentazione

Mediante la gestione delle risorse (lezi.NETResourceManager) e l'editor di corsi (lezi.NETEditor) è stato possibile creare alcuni corsi per capire in primo luogo l'efficacia dell'applicazione nel distribuire contenuti. Questi corsi di prova hanno permesso di ottenere due tipi di feedback differenti: il primo arriva direttamente dalle persone che hanno fruito i corsi di prova, il secondo dagli utenti con i diritti di creators che, una volta archiviate le risorse necessarie, hanno prodotto i corsi.

11.1 Acquisizione e produzione

Il corso di esempio principale pubblicato nel sistema riguarda il Web design. E'un corso svolto al Politecnico di Milano dal professor Paolo Paolini nel 2003. Le lezione avvenivano in videoconferenza. Il professore svolgeva una lezione nella sede di Milano e la sede di Como la seguiva in video conferenza. Per la lezione seguente avveniva il contrario: il professore teneva la lezione a Como e Milano seguiva in remoto, e così via. Durante le lezioni le immagini spedite nell'altra sede venivano comunque registrate su nastro magnetico da una regia locale. Il soggetto inquadrato dalla telecamera era scelto durante la lezione dal professore stesso che poteva in questo modo passare dall'inquadratura del proprio mezzobusto, alle slide che disegnava su un foglio e allo schermo di un PC connesso alla rete. La ripresa delle slide avveniva mediante una telecamera posta sopra e avanti la testa del professore. Nelle due aule, quella con la presenza del professore e quella che seguiva in remoto, erano presenti degli schermi per la visualizzazione delle immagini scelte dal professore. Sul nastro è stato memorizzato ciò che appariva su questi schermi.
Di tutte le video cassette del corso è stata eseguita un acquisizione digitale e una successiva frammentazione dei contenuti in spezzoni diversi. Si è arrivati alla situazione in cui per ogni lezione esisteva il corrispondente file video. Di tutti questi files è stato eseguito l'encoding mediante l'applicazione Windows Media Encoder in modo da produrre dei filmati con estensione WMV compatibili con lo streaming video. L'operazione ha permesso di comprimere i filmati originali sia nella traccia audio che in quella video. L'encoder è stato impostato in modo da avere filmati con 100 Kbps. Lo streaming di filmati aventi tale banda è eseguibile partendo da una linea ISDN o ADSL.

Encoding		
Encoding Video	Windows Media Video V 9	
	Formato	329 x 200
	Frame rate	15 fps
	Image quality	85 %
	Banda	87 kbps
	Key frame interval	8 sec
Encoding Audio	Windows Media Audio V 9	
	Formato	16kHz mono CBR
	Banda	16 kbps

Figura 67: Tabella dei parametri utilizzati durante la fase di encoding dei filmati.

Mediante l'applicazione Windows Media Indexer, alla traccia video e audio è stata aggiunta un'ulteriore traccia, detta traccia di scripting, per pilotare la sincronizzazione delle slides (Vedi Sezione 8.1.3). Sempre mediante la medesima applicazione sono stati aggiunti i markers che, indicando temporalmente gli argomenti presenti nel filmato, permettono lo spostamento veloce dell'esecuzione nel punto corrispondente (Vedi Figura 68).
Oltre all'acquisizione di audio e video sono state salvate le slides disegnate dal professore producendo un file .gif per ogni slide presente. Il professore ha poi messo a disposizione files di documentazione e hyperlinks da allegare alle varie lezioni nell'apposita sezione.
Di tutti questo materiale, filmati, slides e documenti è stato eseguito l'upload mediante il resource manager di lezi.NET organizzando tutto il materiale nelle cartelle del creatore del corso.

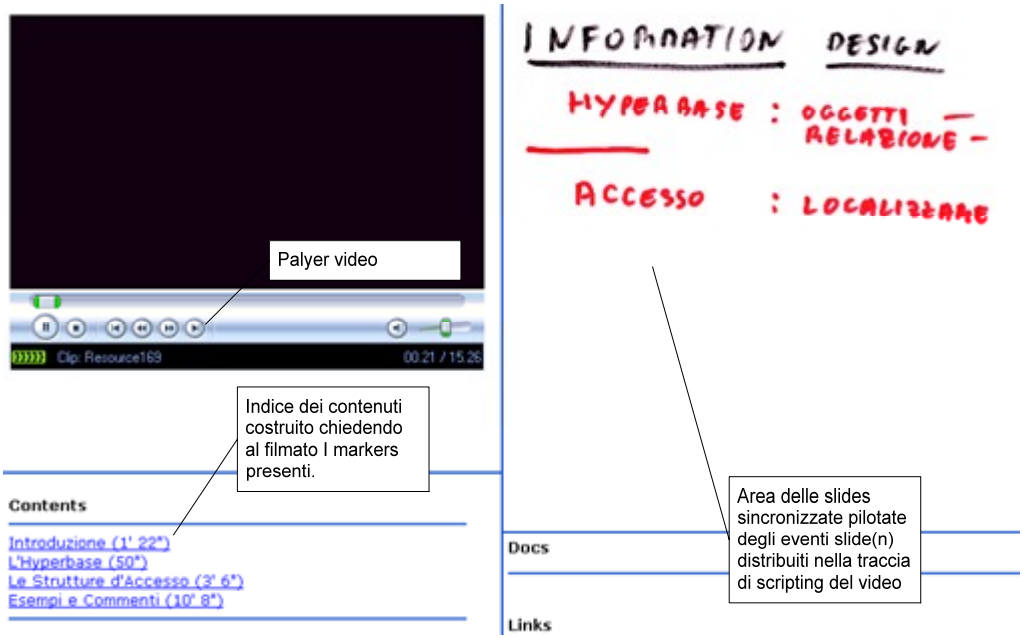


Figura 68: Schermata di una lezione del corso di esempio di Web desing. Cliccando di un marker dei contents si porta l'esecuzione del filmato al punto corrispondente.

Utilizzando infine il lezi.NET editor è stato prodotto e pubblicato il corso Web desing avente la seguente tabella dei contenuti:

- Lezione 1
  - Introduzione: Argomenti del Programma (18' 31")
  - Progetti e Modalità d'Esame (8' 14")
  - Coumnicazione e Didattica (25' 51")
- Lezione 2
  - Introduzione al Design: Il Problema (30' 27")
  - Requisiti e Design Tecnico & Concettuale (15' 17")
  - Il Design Concettuale (32' 9")
  - La Metodologia W2000 (38' 49")
  - L'Information Desing: Introduzione (15' 26")
  - L'Information Design: L'Hyperbase (18' 20")
  - L'Information Design: Strutture d'Accesso (7' 17")

- Lezione 3
  - Design In-The-Large & In-The-Small (14' 7")
  - Scegliere Entity Type e Single Entity (23' 44")
  - Entity Type In-The-Large: Le Componenti (con esempi) (31' 46")
  - Le Associazioni In-The-Large (40' 7")
  - Similitudini & Differenze col Modello E-R (7' 50")
  - Riassunto della Lezione (5' 7")
- Lezione 4
  - Ancora sull'HyperBase (12' 44")
  - Design In-The-Small (25' 42")
  - Le Associazioni Semantiche In-The-Small (10' 19")
  - Presentazione di WebBoard (7' 8")
  - Le Strutture d'Accesso: le Collezioni (con esempi) (58' 17")
- Lezione 5
  - Introduzione al Navigation Design (12' 21")
  - Cluster, Nodi e Pattern Navigazionali (9' 48")
  - I Nodi Navigazionali (30' 38")
  - Navigare le Entità: il Cluster Strutturale (7' 50")
  - Navigare le Associazioni: il Cluster Associativo (29' 44")
  - Naviagare le Collezioni: il Cluster di Collezioni (21')
  - Visita al Sito del National Gallery of Art (NGA) (33' 32")
- Lezione 6
  - Introduzione alla Notazione W2000 (8' 54")
  - Oggetti W2000: Notazioni e Proprietà (18' 12")
  - Entità Singole & Entità Astratte (11' 35')
  - Componenti & Slots (14' 15")
  - Associazioni Semantiche e Centro di Associazione (42' 49")
  - Le Role Entity (14' 42")
  - Le Collezioni e Il Centro di Collezione (21' 50")
  - IL Modello Navigazionale: I Nodi (18' 35")
  - I Cluster Navigazionali (13' 5')
- Lezione 7
  - Commenti su Compitino di Web Design (6' 9")
  - Introduzione al Publishing Desing (24' 3')
  - Metodologia W2000 per il Publishing Design (43' 54")
  - Visita al Sito di Amazon.com (55' 8")
- Lezione 8
  - Segmenti di Preview e di Filtro (8' 15")
  - Notazione per il Modello Navigazionale (43' 26")
  - Domande & Esempi sulla Notazione W2000 (38' 21")
  - Notazione per Navigation Design: In-The-Large & In-The-Small (5' 10")
  - Suggerimenti & Principi di Progettazione (57' 35")

- Lezione 9
  - Relazione su Convegni e Proposte di Progetto su W2000 (29' 1')
- Lezione 10
  - Modifiche al Calendario e Programma della Lezione (11' 8")
  - Progetti d'Esame: Esempi & Proposte (35' 47")
  - Progettazione Realistica di Siti Web: Gli Approcci Tipici (14' 55")
  - La Progettazione Realistica: Multicanalità e Information Design (31' 37")
  - La Progettazione Realistica: Multicanalità, Navigation e Publishing Designs (28' 27")
- Lezione 11
  - Introduzione all'Analisi dei Requisiti per le Applicazioni Web (7' 30")
  - Le Fasi dell'Analisi dei Requisiti: L'Elicitazione (36' 5")
  - Le Distorsioni nell'Elicitazione (11' 52")
  - Dall'Elicitazione all'Analisi: Il Modello AWARE (21' 50")
  - Le Specifiche dei Requisiti (22' 14")
  - Definizione e Tipi di Requisito: le "Dimensioni" dei Requisiti (15' 33")
  - Usare i Requisiti in Fase di Design (3' 17")

Questo corso pubblicato è stato effettivamente seguito dagli studenti del quinto anno del vecchio ordinamento dell'università di Lecce in sostituzione del corso originale di informatica grafica che è stato sospeso.

Un secondo corso di esempio WebTalk Cube, è stato creato per dimostrare la flessibilità dell'ambiente di esecuzione leziNETViewer che è in grado di mandare in esecuzione contenuti didattico di qualsiasi tipo. In Figura 69 è visibile una cattura video di un corso multimediale che utilizza un motore di rendering tridimensionale per creare un ambiente virtuale in cui effettuare esperienze di learning collaborative.

Figura 69: Corso WebTalk Cube.

In questo corso il lezi.NET viewer è in gradi di integrarsi con il server di WebTalk che gestisce la collaborazione fra le diverse istanze degli avatar degli utenti. Da notare che nonostante la complessità tecnica del corso la piattaforma lezi.NET è in gradi di tracciare le prestazioni dell'utente e mantenere la storia della fruizione.

Un ulteriore esempio di utilizzo di lezi.NET riguarda la sua integrazione come visualizzatore di learning objects atomici all'interno dell'ambiente Virtual Campus. Il corso è gestito quindi da Virtual Campus è riguarda un corso di Software design. I vari LOs di cui è composto questo learning object complesso (CLO) sono mandati in esecuzione attraverso lezi.NET il quale si appoggia al server di WebTalk Cube poiché si tratta anch'essa di un'esperienza collaborativa in ambiente tridimensionale. Con questo esempio è stato possibile dimostrare la possibilità di integrazione delle diverse piattaforme.

11.2 Impressioni d'uso

Le impressioni di utilizzo da parte di utenti, che hanno utilizzato questo corso di esempio, sono state essenzialmente positive. Gli utenti hanno evidenziato come il sistema sia utile, oltre che per fruire la prima volta il corso, anche per avere una sorta di "reference" da consultare durante lo studio dell'esame E' stata apprezzata la possibilità di poter scaricare la documentazione relativa ad una lezione in modo veloce e semplice. Grazie al contenuto multimediale (audio, video e slides sincronizzate) l'utente ha un'esperienza di learning simile a quella che si ha durante la frequenza di una lezione in video-conferenza, con in più il vantaggio di rivedere i passaggi di una lezione che non vengono immediatamente compresi. Lo svantaggio è, essendo l'esperienza asincrona, l'impossibilità per l'utente di rivolgere immediatamente delle domande al docente. Tuttavia tale limitazione è facilmente aggirabile mediante un sistema di messaggistica quale la posta elettronica, o meglio, un forum integrato nell'applicazione stessa tramite il quale poter scambiare anche impressioni e opinioni sul corso con gli altri utenti. Un estensione di questo tipo sarebbe molto apprezzata dagli utenti.

Gli utenti che producono i contenuti hanno dato un opinione positiva sul gestore delle risorse e sull'editor di corsi. Apprezzata è stata la possibilità di riutilizzare ed eventualmente condividere con altri utenti creatori le risorse del proprio archivio per la creazione di corsi. Durante la fase di apprendimento della produzione di un corso, le maggiori difficoltà sono state incontrate durante l'utilizzo delle applicazioni per effettuare l'encoding e l'indexing dei filmati. Dalle operazioni di upload dei contenuti in poi, gli utenti non hanno incontrato ulteriori difficoltà. Estensioni dell'applicazione atte a semplificare la fase di pre-produzione di un corso (acquisizione, encoding e indicizzazione) potrebbero essere quindi un ulteriore sviluppo di lezi.NET.

Oltre agli aspetti precedentemente trattati durante l'utilizzo sono inoltre stati fatti i seguenti suggerimenti per eventuali modifiche/estensioni del sistema.

Lato fruizione:

- Implementazione di un efficace motore di ricerca che riesca a ricercare parole chiave anche fra i markers dei filmati.
- Implementazione di un "segnalibro" indicante il punto esatto in cui si era interrotta l'ultima volta la fruizione per velocizzare i successivi accessi.
- Indice del materiale (Documenti, slides etc) allegato al corso.
- Possibilità di accedere ad una form pubblica per la richiesta automatica di un account.

Lato editing corsi:

- Sistema per effettuare upload multipli di risorse. Ad esempio caricare sul sistema, compilando una sola form, un insieme di slides. Questo per evitare di compiere operazione ripetitive e tediose.
- Possibilità oltre di riutilizzare oltre che raw data anche SCO ed eventualmente pezzi di corso, dando la possibilità di assemblare i diversi pezzi per produrre diversi corsi.
- Possibilità di creare corsi composti da parti il cui accesso è regolato dall'appartenenza ad un gruppo. Ad esempio, un corso potrebbe avere delle parti dedicate ad un utente avanzato. Tali parti dovrebbero essere eseguibili solo dagli utenti appartenenti anche al gruppo CorsoXAdvancedUser. Gli altri utenti vedrebbero il corso come se le sezioni avanzate non esistessero.

12 Conclusioni

La progettazione e la realizzazione del progetto ha permesso di ottenere un sistema “vicino “ per funzionalità ad un prodotto commerciale. Terminate la fase di produzione e quella di testing e dopo un utilizzo di alcuni mesi da parte di diversi utenti è possibile fare il seguente bilancio:

- **Raggiungimento degli obiettivi preposti**: Tutti gli obiettivi di cui si è parlato nell’introduzione e durante l’analisi dei requisiti sono stati raggiunti. Durante la fase di progettazione e quella implementativa sono state aggiunte diverse nuove funzionalità in quanto ci si è accorti della necessità di alcuni strumenti utili per rendere il sistema più usabile. Il componente lezi.NETViewer è stato integrato con successo nell’ambiente Virtual Campus. Il sistema di workflow che regola il sequencing dei LOs è stato in grado di eseguire learning objects atomici all’interno del viewer. E’ stato possibile inoltre integrare la piattaforma lezi.NET con il sistema WebTalk Cube in grado di fornire un ambiente di fruizione di contenuti didattici tridimensionali e collaborativi. Si è inoltre verificato che durante la produzione di corsi con gli strumenti messi a disposizione del lezi.NETEditor vi è stata una sensibile riduzione del tempo di produzione di un corso e quindi del relativo costo.

- **Utilizzo di standard di riferimento per l’e-learning**. L’utilizzo di IMS e SCORM ha permesso di creare un learning management system completo che ha superato i test di validazione per SCORM 1.2. Questi standard non devono però avere la pretesa di essere delle specifiche formali riguardanti la realizzazione di un LMS. Sono, infatti, molte le parti non chiare nei diversi documenti di specifica. Leggendo sui forum di ADL e IMS è possibile capire quali siano le intenzioni dei creatori di questi standards. Le specifiche sono in continua evoluzione, mancanze ed inconsistenze sono continuamente corrette, versione dopo versione. Le diverse specifiche devono essere quindi considerate come delle linee guida che può essere molto utile seguire durante la realizzazione di un LMS.

- **Produzione di corsi di test e validazione significativi**. Il corsi creati, di cui si è parlato nella Sezione 11, hanno permesso di dimostrare l’efficacia e l’usabilità dell’applicazione.

- **Testing e utilizzo della nuova piattaforma di Microsoft .NET**. La valutazione su questo aspetto, soprattutto tenendo conto del fatto che il framework utilizzato è la versione 1.0, è sicuramente positiva. Non si sono mai verificati inattesi malfunzionamenti delle classi di librerie del framework. Anche il server Web (IIS + Worker process ASP.NET) si è dimostrato molto stabile permettendo, durante la fase di sviluppo, di concentrarsi esclusivamente sull’applicazione. E’ da notare il fatto che lezi.NET e’ al momento l’unico LMS interamente sviluppato con tale tecnologia.

Le possibili estensioni del sistema riguardano lo sviluppo del lezi.NETViewer al fine di soddisfare i requisiti descritti dal nuovo standard SCORM 1.3 che è attualmente in fase di definizione. La maggiore novità di tale evoluzione riguarda la standardizzazione dei meccanismi di workflow per poter istruire un eventuale motore di sequencing per gli SCO. L’architettura flessibile della piattaforma lezi.NET dovrebbe permettere di implementare

questa nuova importante caratteristica in modo relativamente semplice. In pratica si tratta di inserire nell'architettura di Figura 35 il sequencer di SCO indicato nell'architettura generale di Figura 1.

Una funzionalità da integrare sicuramente in una eventuale futura versione di lezi.NET è una strumento che permetta di fare della reportistica in modo da poter avere una visione chiara ed intuitiva delle informazioni riguardanti le prestazioni degli studenti raccolte durante la fruizione del corso dal tracking service. Sfruttando questo componente potrebbe essere possibile anche introdurre un modulo per la valutazione automatizzata della didattica. Il tutor o il docente, configurate particolari politiche di valutazione nel modulo di valutazione, potrebbe ottenere direttamente una lista di valutazioni per gli utenti del corso pubblicato su lezi.NET.

Un'altra funzionalità che sarebbe interessante integrare è un forum di discussione avanzato con la possibilità di condividere materiale didattico da parte degli studenti oltre che di scambiare opinioni, fare domande etc e una servizio di messaggistica immediata per rendere ancora più interattiva l'esperienza di learning.
Altre possibili estensioni e sviluppi futuri sono stati direttamente suggeriti dagli utenti. E' possibile vederne una sintetica lista nella Sezione 11.2.

Bibliografia e riferimenti

ADL: Advanced Distributed Learning: http://www.adlnet.org.

ADL.SCORM.OW: SCORM Overview Ver 1.2: http://www.adlnet.org.

ADL.SCORM.CAM: SCORM Content Aggregation Model Ver 1.2: http://www.adlnet.org

ADL.SCORM.RTE: SCORM Run-Time environment Ver 1.2: http://www.adlnet.org.

ADL.SCORM.RTE.DM : SCORM Run-Time environment Ver 1.2: Data Model: http://www.adlnet.org.

AICC-CMI CMI001 Guidelines for Interoperability Version 3.4. October 23, 2000. Include: AICC Course Structure Format, AICC CMI Data Model: http://www.aicc.org/.

BLACKBOARD: Blackboard: http://www.blackboard.com

CDES: Allen, P., Frost, S. and Yourdon E.**,** *Component-based development for enterprise systems": Applying the Select Perspective*, Managing Object Technology Series, No 12, Cambridge University Press, 1998

CENTRA: Centra: http://www.centra.com

C2L: Click2Learn: http://www.click2learn.com

CMVC: Van Gorp, Mark and Boysen, Pete. *ClassNet: Managing the Virtual Classroom*, International Journal of Educational Telecommunications, Vol. 3(2/3), 279 - 292 , 1997

COL: Corsi On Line: http://corsi.metid.polimi.it

CSWLE: a Tool for the Creation of Sophisticated Web-Based Learning Environments : Murray W. Goldberg and Sasan Salari, Department of Computer Science, University of British Columbia: An Update on WebCT (World-Wide-Web Course Tools) -

DPL: DataPower Learning: http://www.datapower.co.uk

DSWC: *Using a Web-Based Course Authoring Tool to Develop Sophisticated Web-Based Courses*: Murray W. Goldberg, Department of Computer Science, University of British Columbia, Vancouver, British Columbia, Canada:

DOCENT: Docent: http://www.docent.com

EBC: An Environment for Building WWW-Based Courses: Murray W. Goldberg, Sasan Salari, Paul Swoboda: World Wide Web - Course Tool: - Fifth International World Wide Web Conference - May 6-10, 1996, Paris, France

EEBI: Exploiting an event-based infrastructure to develop complex distributed systems: G. Cugola, E. Di Nitto, and A. Fuggetta.. In Proc. of the 19th Int. Conf. on Software Engineering (ICSE98), 1998.

EDUPRISE: Eduprise: http://www.eduprise.com

EPK: The Effects of Prior Knowledge and Goal Strength on the Use of Hypertext : David A. Last and Angela M. O'Donnell, Rutgers, The State University of New Jersey: - Journal of Educational Multimedia and Hypermedia (2001) 10(1), 3-25

EUL: EU eLearning: http://europa.eu.int/comm/education/elearning/

EFST: A Formative Evaluation of Scenario: Hope D. Harley, Cheryl D. Seals, Mary Beth Rosson: Based Tools for Learning Object-Oriented Design (1998)

GK: Global Knowledge: http://www.globalknowledge.co.uk

IPREP: Intelliprep: http://www.intelliprep.com

LOGILENT: Logilent: http://www.logilent.com

RKWLO: Clark, R.C., “Recycling knowledge with learning objects” Training and development, Vol 52, No 10., Oct, 1998, pp. 60-63.Instructional Design: International Perspectives, Vol . 2, Mahwah, NJ

ARIADNE: Alliance of Remote Instructional Authoring and Distribution Networks for Europe: http://www.ariadne-eu.org/)

ECMA.CLI: Common Language Infrastructure. International Standard: http://www.ecma-international.org/publications/standards/ecma-335.htm

ECMA.C#: C#. International Standard: http://www.ecma-international.org/publications/standards/ecma-334.htm

ERDDA: Enabling the Rapid Development of Dependable Applications in the Mobile Environment. A. Murphy.PhD thesis, Washington University in St. Louis, MO, USA, Aug. 2000.

IEEE: : IEEE Information Technology : http://www.ieee.org

IEEE.LTSA: Learning Technology Systems Architecture (LTSA): P 1484.1/D9, 2001-11-30 Draft Standard for Learning Technology-, http://edutool.com/ltsa

IEEE.LOM: IEEE Information Technology - Learning Technology - Learning Objects Metadata: LOM: Working Draft 6.1 (2001-04-18).: http://ltsc.ieee.org/

IMS : IMS Global Learning Consortium Inc.: http://www.imsglobal.org/

IMS.CP: IMS Content packaging specification 1.1.2: IMS Global Learning Consortium, Inc.: http://www.imsglobal.org/

LTPI: Linking and Timing Information Presentation in Multimedia Educational Systems: Amelia K.Y. Tong, MODE International, The Media Centre: – Journal of Educational Multimedia and Hypermedia (2001) 10(2), 185-203

MA: Macromedia Authorware: http://www.macromedia.com/software/authorware/

MAET: Schank, Roger C., Michael Korcuska, and Menachem Jona, 1995: Multimedia Applications for Education and Training: Revolution or Red Herring?, ACM Computing Surveys, Vol. 27, Number 4, 633-635.

MC: Metacollege: http://www.metacollege.com

MIMICT: Modeling Instruction with Modern Information and Communications Technology: the Mimic Project :Ronald Abate, Cleveland State University, USA; Jennifer Cutler-Merritt, John Carroll University, USA; James Meinke, Baldwin Wallace College, USA; Mary Jo Cherry, Ursuline College, USA; David Shutkin, John Carroll University, USA:– Society for information technology & teacher education – 2001 12th International Conference

MS.EL: Microsoft eLearn: http://www.microsoft.com/elearn/

MS.NET.FRAMEWORK: Microsoft .NET Framework: http://msdn.microsoft.com/netframework

MVC: model – view – controller: design pattern.: http://www.cs.indiana.edu/~cbaray/projects/mvc.html

MVLC: A Model for Virtual Learning Communities in the Learning Communities in the World Wide Web: Avigail Oren, Rafi Nachmias, David Mioduser, And Orly Lahav – Tel-Aviv University – School of Education – Israel: Learnet — International JI. Of Educational Telecommunications (2000) 6(2), 141-157

OIL: Oracle Ilearning: http://www.oracle.com/ilearning

PDOI: Principles for Designing Online Instruction: Harriett Bohannon, Florida Gulf Coast University, USA; Peggy Bradley, Florida Gulf Coast University, USA; Joan Glacken, Florida Gulf Coast University, USA; Roberta McKnight, Florida Gulf Coast University, USA:– Society for information technology & teacher education – 2001 12th International Conference

SIES: Roschelle, J., Kaput, J., Stroup, W. and Kahn, T.M., "Scalable integration of educational software: exploring the promise of component architectures", Journal of Interactive media in education, volume 6, 1998, www-jime.open.ac.uk/98/6

SSCLI : Shared Source Common Language Infrastructure. Rotor project.

SECURE: Writing Secure Code

TDCMI: The Dublin Core Meta-Data Initiative. http://purl.oclc.org/dc/groups/index.htm

THE: Internet-Based Testing: A Vision or Reality?: Eleanor Bicanich, Dr. Thomas Slivinski, Dr. Susan B. Hardwicke, Dr. Jerome T. Kapes: The journal ONLINE – Technological Horizons in Education

TJEBI: The **jedi** event-based infrastructure and its application to the development of the opss wfms: G. Cugola, E. Di Nitto, and A. Fuggetta. IEEE Transactions on Software Engineering.

TVS: The virtual school: an integrated collaborative environment for the classroom: Isenhour (et al.) 2000. Educational Technology & Society 3(3) 2000

WBTS: WBT System: http://www.wbtsystems.com

W3C.SOAP: Simple Object Access Protocol: http://www.w3.org/TR/2003/PR-soap12-part1-20030507/, http://www.w3.org/TR/2003/PR-soap12-part2-20030507/

W3C.WSDL: Web Services Description Language: http://www.w3.org/TR/wsdl12/, http://www.w3.org/TR/wsdl12-bindings/

W3C.UDDI: Universal Description, Discovery and Integration: http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

WM.WME: Windows Media, Windows Media Encoder : http://www.microsoft.com/windows/windowsmedia/wm7/encoder.aspx

Appendice A

Per poter effettuare delle prove in modo da conoscere e familiarizzare con lezi.NET è possibile accedere al sistema mediante l'URL: http://hoc104.elet.polimi.it/lezinet utilizzano l'utente dimostrativo con ruolo di Creator.

UserID: **demo**
Password: **demo**

Il seguente manuale utente è una versione sintetica del manuale completo in primo luogo per evitare che le sezioni seguenti risultino essere eccessivamente voluminose e in secondo luogo poiché questo documento non ha come scopo principale quello di essere un manuale utente che spieghi passo per passo il funzionamento della piattaforma nella sua interezza. Le sezioni seguenti sono infatti inserite con il solo scopo di permette al lettore del presente documento di orientarsi nell'applicazione in modo da poterne apprezzare appieno le funzionalità.

Manuale utente

Login

Per poter effettuare l'accesso all'applicazione è necessario essere in possesso di un account composto da userID (identificativo utente) e password. Contattare un amministratore o autore di corsi per ottenere un account.
Se in possesso di credenziali valide inserire userID e password nella form di login. Vedi Figura 70.

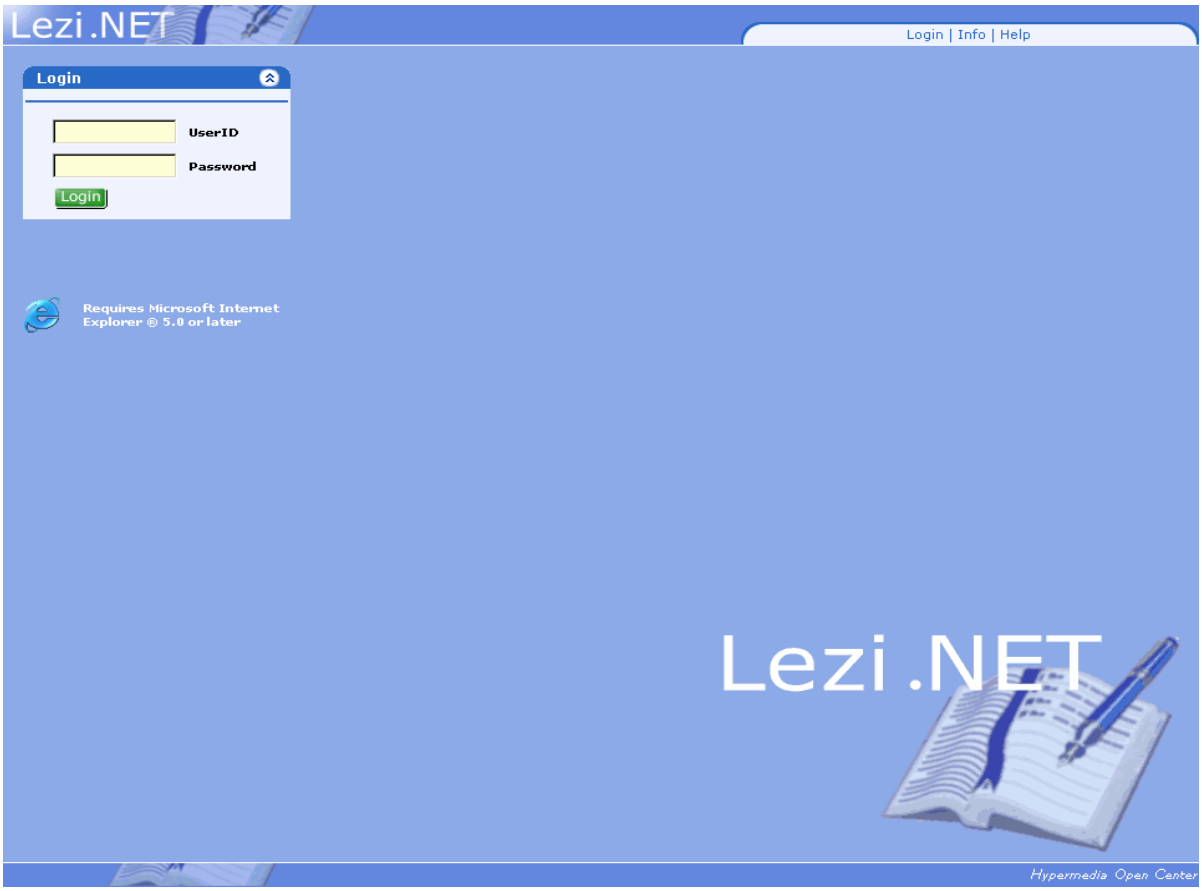


Figura 70: Pagina di login per effettuare l'autenticazione.

Premere il pulsante “Login” per accedere.

Start

Questa è la pagina che si presenta dopo aver effettuato l’accesso all’applicazione. Vedi Figura 71.

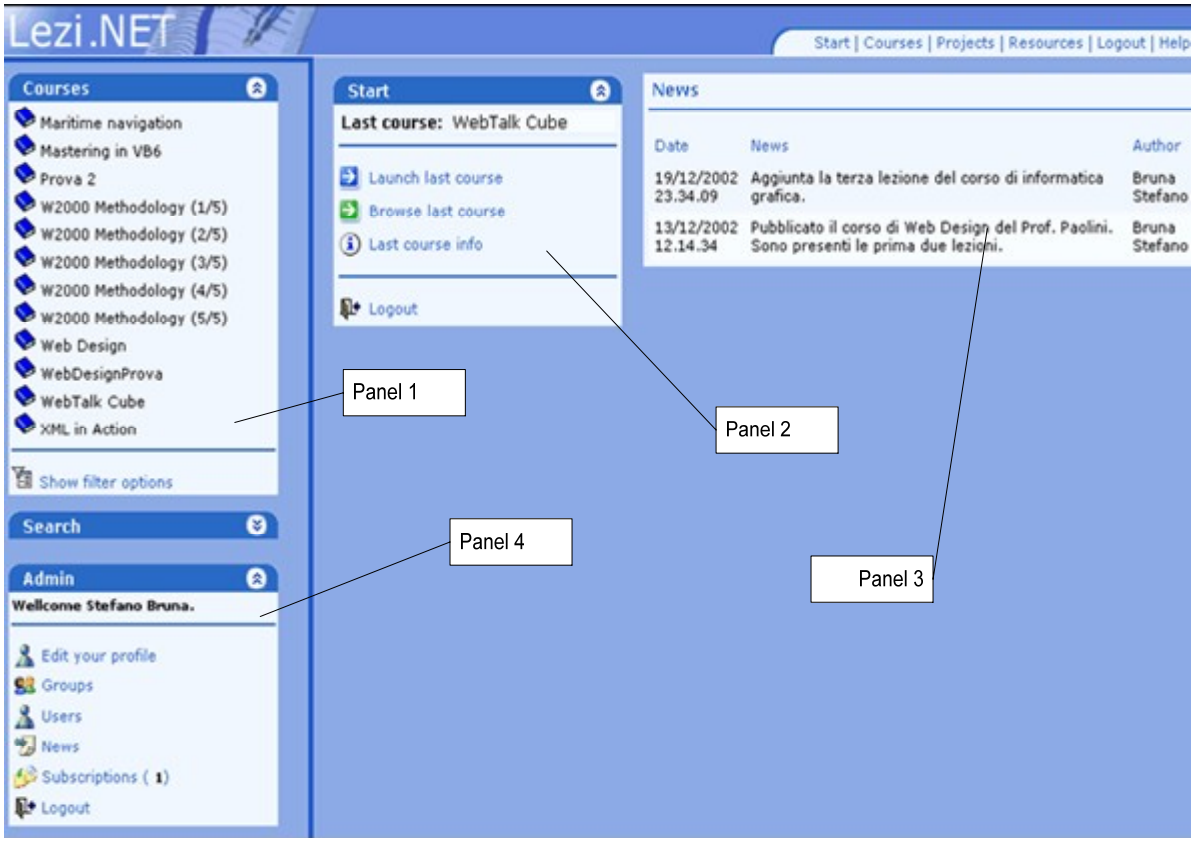


Figura 71: Pagina iniziale Start

Nel Panel 1 (Courses) è possibile avere una lista di corsi filtrata e ordinata in base ai criteri selezionati.
E' possibile modificare tali criteri aprendo il un pannello premendo sul link 2. Utilizzando questo pannello si possono visualizzare i corsi in base a:

- **Published courses**: corsi pubblicati.
- **Published courses on server**: corsi pubblicati solo presso il server sul quale sia ha effettuato l'autenticazione.
- **Not subscribed courses**: corsi per i quelli non si hanno diritti di frequenza.
- **Preferred courses**: corsi memorizzati nella lista dei propri preferiti.
- **Subscribed courses**: corsi per i quali si hanno diritti di frequenza.

E' possibile, inoltre, stabilire l'ordinamento dei corsi in base a:

- **By name**: ordinamento alfabetico crescente basato sul nome del corso.
- **By date (publishing)**: ordinamento crescente in base alla data di pubblicazione.
- **By date (creation)**: ordinamento crescente in base alla data di creazione.

Cliccare sul nome di un corso per selezionarlo. (vedi sezione 1.5).
Nel Panel 2 di Figura 71 è possibile avere due link veloci per accedere direttamente all'ultimo corso frequentato oppure alla pagina delle informazioni sempre dell'ultimo corso frequentato.
Tramite il pulsante "Logout" dello stesso pannello è possibile uscire dall'applicazione.

Il pannello 3 di Figura 71 visualizza una lista di news pubblicate degli autori di corsi a dagli amministratori di sistema. Premendo sui nomi delle colonne è possibile ordinare le voci in base a qual criterio. Ad esempio premendo su "Author" è possibile ordinare le news in base al nome dell'autore.

Profilo

Navigando tramite il link "Edit your profile" posizionato nel Panel 4 di Figura 72 è possibile raggiungere la pagina riguardante informazioni relative al profilo.

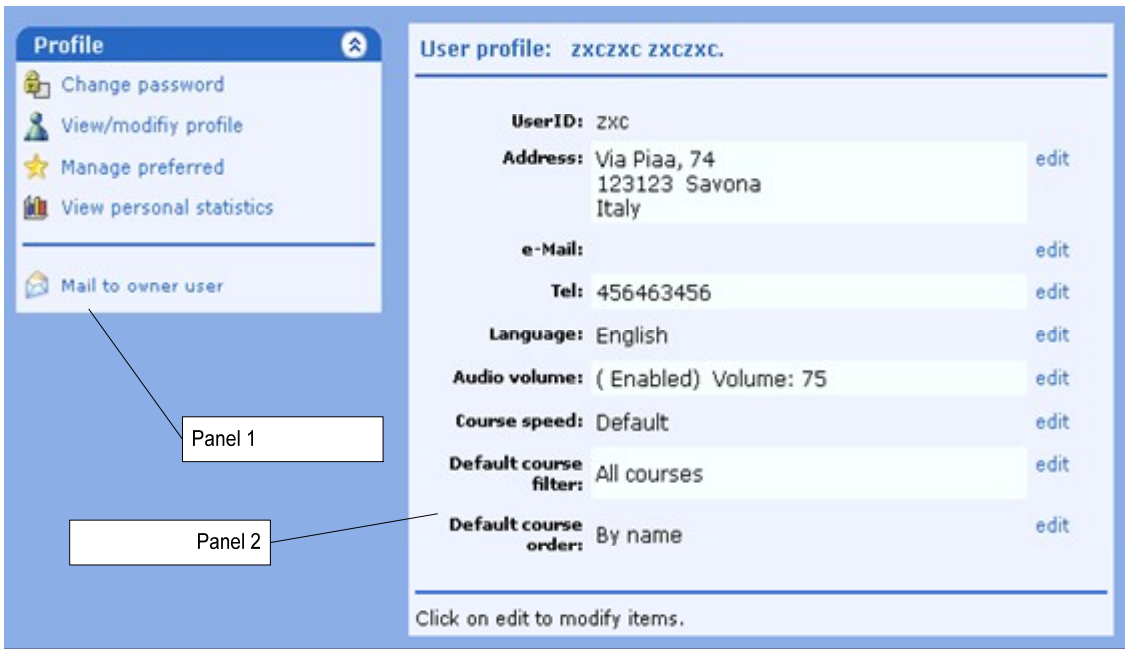


Figura 72: Sezione della pagina per la personalizzazione del proprio profilo utente.

Nel panel 2 di Figura 72 sono rappresentate le informazioni riguardanti l'utente. E' possibile modificare il proprio indirizzo, la propria e-Mail, il proprio telefono, il linguaggio per eventuali corsi predisposti all'utilizzo con diverse lingue, il volume dell'audio per i corsi multimediali, la velocità del corso, il filtro e il criterio di ordinamento di default per la lista dei corsi del Panel 1 di Figura 71.
Premendo il link "edit" corrispondente ad una voce è possibile attivare la modalità di editing. Ad esempio premendo "edit" in corrispondenza di indirizzo si ottiene la form visibile in Figura 73.

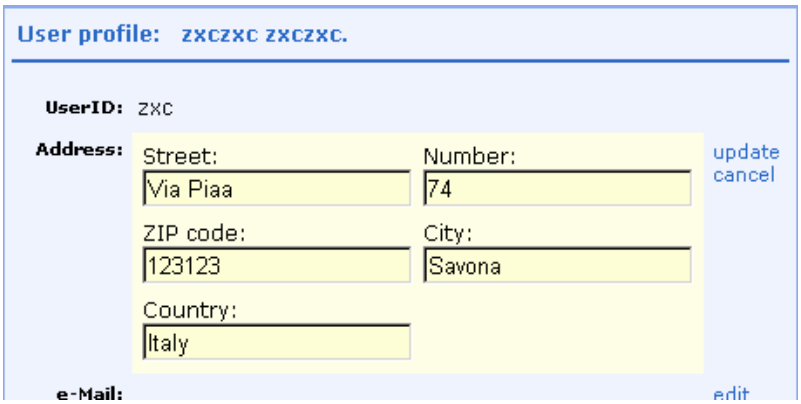


Figura 73: Modalità di editing del campo Address.

Premendo sul link "cancel" di Figura 73 si esce dalla modalità di editing del campo address senza confermare eventuali modifiche. Se invece si preme su "update" le modifiche apportate vengono salvate. Lo stesso criterio è utilizzato anche per gli altri campi.
Nel panel 1 di Figura 72 sono presenti le seguenti voci:

- **Change password**: Permette di modificare la propria password a patto di conoscere quella vecchia.
- **Manage preferred**: Permette di gestire la lista dei propri corsi preferiti. Modificando eventualmente il nome indicativo del corso.
- **View personal statistics**: Visualizza la data di creazione dell'account, il numero di accessi eseguiti dalla data di creazione, la data dell'ultimo accesso e l'ultimo corso visto.
- **View modify profile**: Rivisualizza la form iniziale per la modifica del profilo.

- **Mail to owner user**: apre il client di posta elettronica di default dell'utente per la spedizione di una mail all'utente che ha creato l'account.

Corsi

La pagina dei corsi propone una vista, anch'essa filtrabile, della lista dei corsi presenti nel sistema come per il Panel 1 di Figura 71. Sono presentate in forma tabellare titolo del corso, abstract, data di creazione e autore. Vedi pannello 2 di Figura 74.

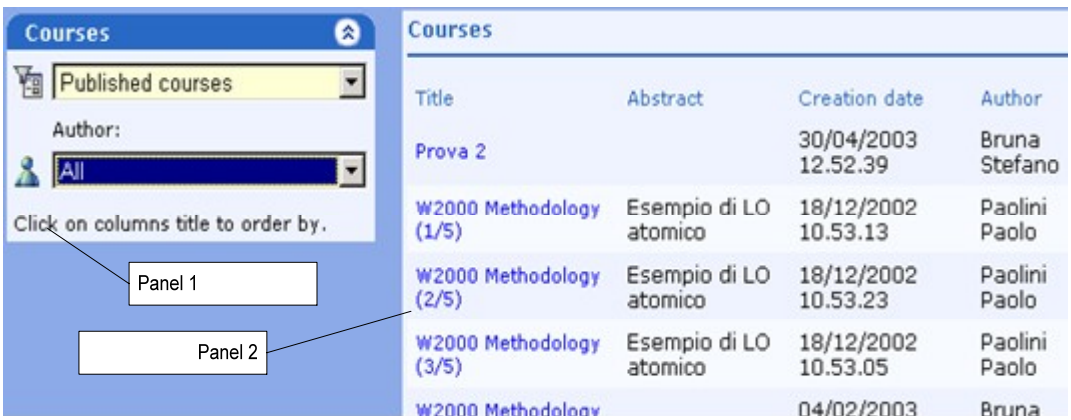


Figura 74: Pagina Corsi. E' possibile filtrare la lista dei corsi anche in base all'autore del corso. Premendo sui titoli delle colonne è possibile ordinare in modo crescente / decrescente la lista.

Corso

La pagina del corso presenta visualizza alcuni pannelli che visualizzano informazioni e propongono una lista di operazioni fattibili dall'utente relative al corso. Nella parte destra della pagina è presentata la tabella dei contenuti del corso. Vedi Figura 75.

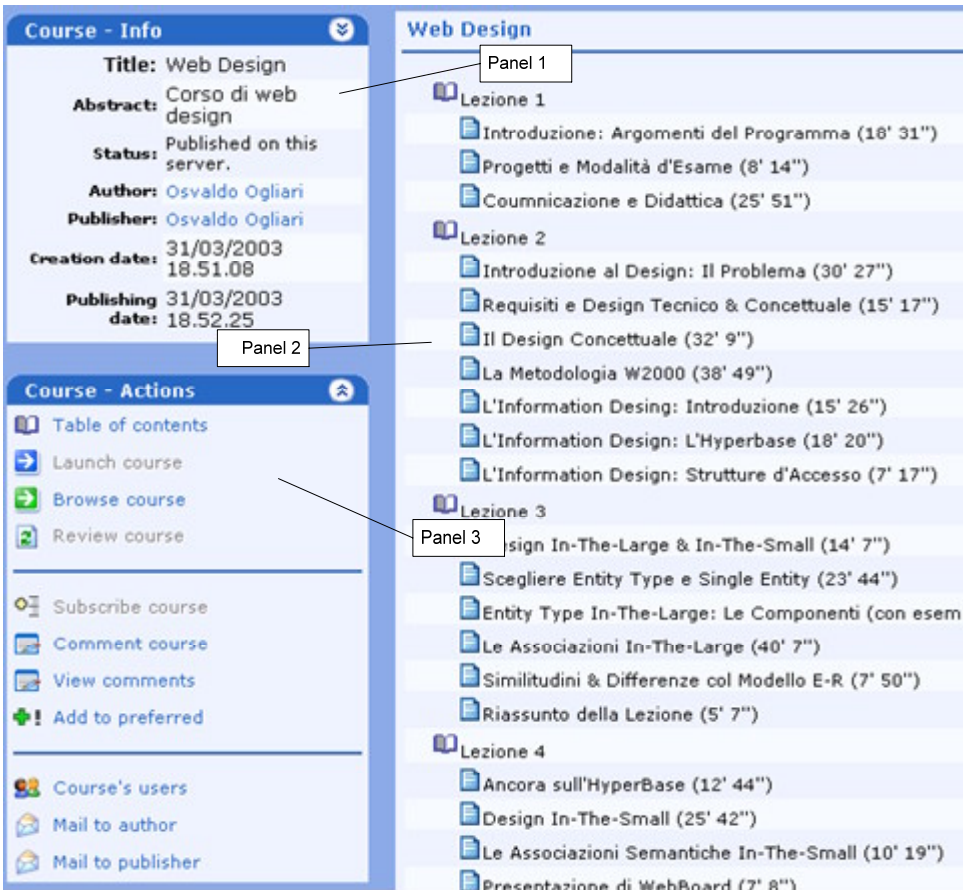


Figura 75: Pagina del corso di Web design.

Nel Panel 1 di Figura 75 sono visibile alcuni dati relativi al corso. Nel pannello 3 invece è presente una lista di operazioni relative al corso visualizzato. Le voci di tali lista sono abilitate quando il carattere è grigio e abilitate quando il link è attivo e il carattere è azzurro. L'abilitazione o disabilitazione di una voce è regolata dal ruolo[40] dell'utente, dallo stato del corso e dai diritti che l'utente ha sul corso. Sempre in base al ruolo dell'utente alcune voci non sono addirittura visibili in quanto tali operazioni non saranno mai eseguibili dall'utente a meno che non assuma un ruolo con maggiori possibilità. Le azioni eseguibili sono :

- **Table of contents**: Ritorna a visualizzare la tabella dei contenuti nel Panel 2 di Figura 72. Tale pannello può essere infatti sostituito da form, messaggi e altre informazioni durate l'interazione dell'utente con la pagina.

[40] *Con ruolo si intende la possibilità da parte di un utente di eseguire determinate operazioni caratteristiche di una o più categorie predefinite a cui l'utente appartiene direttamente o indirettamente. Ad esempio, se l'utente appartiene al gruppo predefinito Creators avrà la possibilità di accedere alle sezioni dell'applicazione in cui è possibile creare un corso. L'appartenenza indiretta si ha,ad esempio, quando un utente appartiene ad un gruppo che sua volta appartiene ad un gruppo predefinito. Per maggiori dettagli vedere Sezione 6.1*

- **Launch course**: Esegue un corso. Se il corso è predisposto[41], registra l'interazione utente e mantiene uno stato di esecuzione. Affinché tale voce sia abilitata l'utente deve avere i diritti di esecuzione sul corso e il corso stesso deve poter essere eseguibile in tale modalità.
- **Browse course**: Permette la navigazione di un corso senza tenere traccia delle azioni dello studente.
- **Review course**: Permette di vedere un corso e lo stato raggiunto all'ultimo Launch senza, tuttavia, modificarlo.
- **Subscribe course**: Premendo questo link è possibile inoltrare una richiesta di sottoscrizione, compilando un form, al creatore del corso e/o all'amministratore di sistema. Tale voce è abilitata solo se non si ha alcun tipo di diritto su corso. Per poter fare la richiesta di sottoscrizione è necessario compilare una form che richiede la motivazione della richiesta.
- **View comments**: Mediante questa voce è possibile ottenere una lista di commenti che gli altri utenti hanno fatto sul corso.
- **Add to preferred**: Link per poter aggiungere il corso corrente alla propria lista di corsi preferiti.
- **Course's users**: Visualizza un elenco di utenti che hanno diritti almeno in lettura sul corso corrente. In questo modo è possibile vedere quali sono gli utenti che possono fruire del corso.
- **Mail to author**: apre il client di posta elettronica di default dell'utente per la spedizione di una mail all'utente che ha creato il corso.
- **Mail to publisher**: apre il client di posta elettronica di default dell'utente per la spedizione di una mail all'utente che ha pubblicato il corso.

I comandi Launch Course, Browse Course e Review Course causano l'apertura del viewer dei corsi.

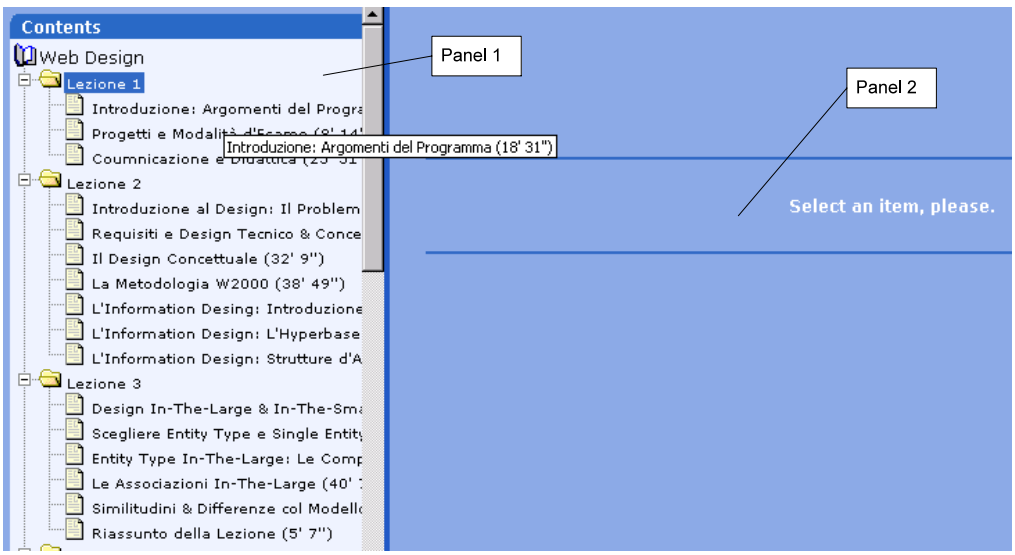


Figura 76: Pagina del viewer. E' selezionato il corso di Web design. Per poter visualizzare completamente la tabella dei contenuti è possibile ridimensionare le dimensioni dei Panel.

[41] *Appartiene allo standard SCORM*

Nel Panel 1 di Figura 76 è presente la tabella dei contenuti del corso. Selezionando una voce da tale elenco, il corrispondente contenuto sarà visualizzato nel Panel 2. In basso sono presenti due pulsanti per selezionare il contenuto precedente o quello seguente. Mediante il link “Close course” è possibile terminare la fruizione ritornando alla pagina del corso. Con il link “Close outline” è possibile nascondere e visualizzare la tabella dei contenuti. Tale comando è comodo per permettere la visione quasi a schermo intero.

Appendice B

Manuale utente creatore

Il manuale dell'utente creatore estende quello dell'utente normale in quanto l'applicazione quando si ha il ruolo di Creator permette l'accesso a tutte le aree dell'utente normale è in più quelle dedicate alla gestione delle risorse, la gestione dei gruppi e degli utenti e la gestione dei progetti.

Groups

Per poter accedere alla gestione dei gruppi è necessario premere sul link Groups del Panel 4 di Figura 71. L'utente con il ruolo di Creator potrà modificare esclusivamente la composizione dei gruppi di propria appartenenza. Può, invece, aggiungere ad un proprio gruppo un utente di cui non è proprietario. Il Creator può naturalmente modificare solo la composizione dei gruppi di cui è proprietario.

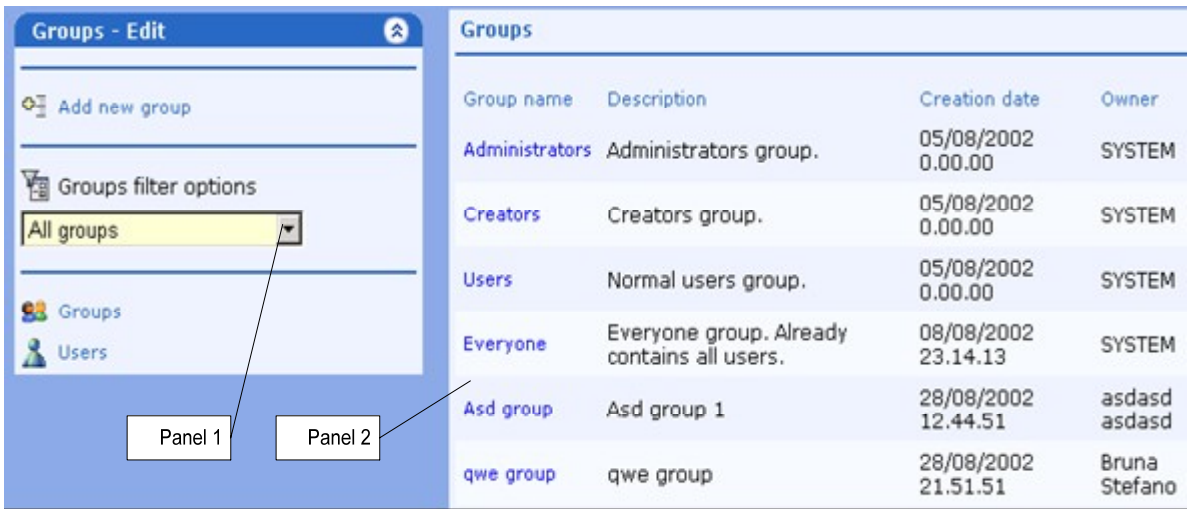


Figura 77: Gestione dei gruppi.

In Figura 77 è possibile vedere il layout della pagina di gestione dei gruppi. Nel Panel 1 è presente un menu a tendina per poter filtrare la lista di gruppi presente nel Panel 2. Con il link "Add new group" è possibile visualizzare nella posizione del Panel 2 una form per l'inserimento dei dati relativi ad un nuovo gruppo. Premendo sulle intestazioni delle colonne del Panel 2 è possibile imporre un ordinamento crescente/decrescente.

Premendo sul nome di un gruppo è possibile visualizzare la pagina del gruppo selezionato con le sue informazioni. In Figura 78 è stato selezionato, ad esempio, il gruppo Administrators. Nel Panel 1 sono presenti le azioni eseguibili sul gruppo, nel Panel 2 sono visibili delle informazioni e, tramite il link "edit" è possibile eseguirne la modifica. Infine nel Panel 3 e 4 sono elencati rispettivamente gli utenti e i gruppi appartenenti al gruppo selezionato.

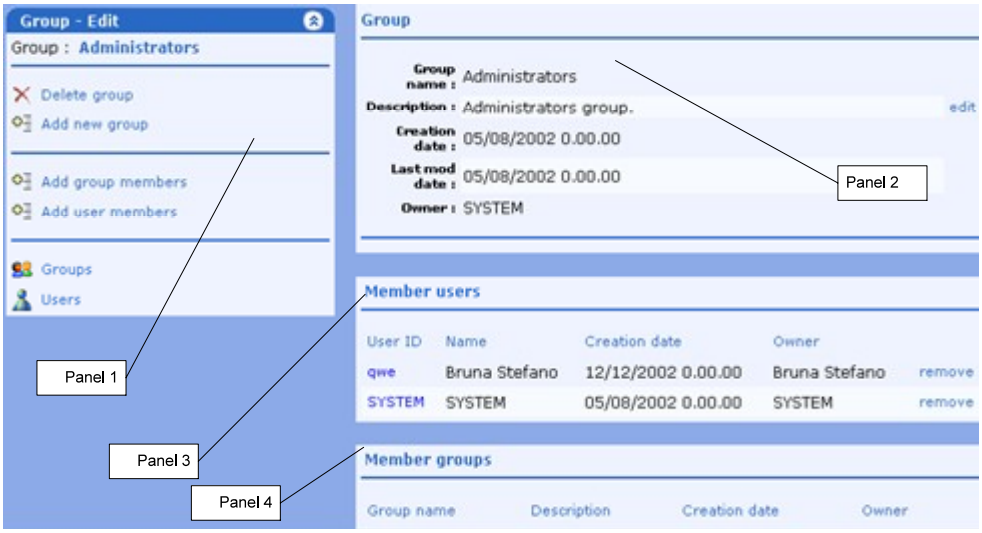


Figura 78: Pagina di un gruppo.

In Figura 79 è possibile vedere un esempio di lista di utenti che può apparire quando si preme il link “Add user member” del Panel 1 di Figura 78. Selezionando uno o più utenti spuntando le voci e premendo il pulsante “Submit” è possibile aggiungere gli utenti selezionati nel gruppo corrente. Premendo sulla userID dell’utente si naviga verso la pagina con le informazioni dell’utente.

Figura 79: Selezione di utenti per farli diventare membri di un gruppo.

Users

La gestione degli utenti per l’utente Creatore è del tutto analoga a quella dei gruppi. Si ha la possibilità, come già precedentemente sottolineato, di poter rendere membro di un proprio gruppo anche un utente di cui non si è proprietario. E’ tuttavia possibile modificare solo i propri utenti. In Figura 80 è possibile vedere parte della pagina di gestione degli utenti. Nel Panel 2 è presente un filtro che permette di discriminare fra utenti di cui si è proprietario e utenti del sistema.

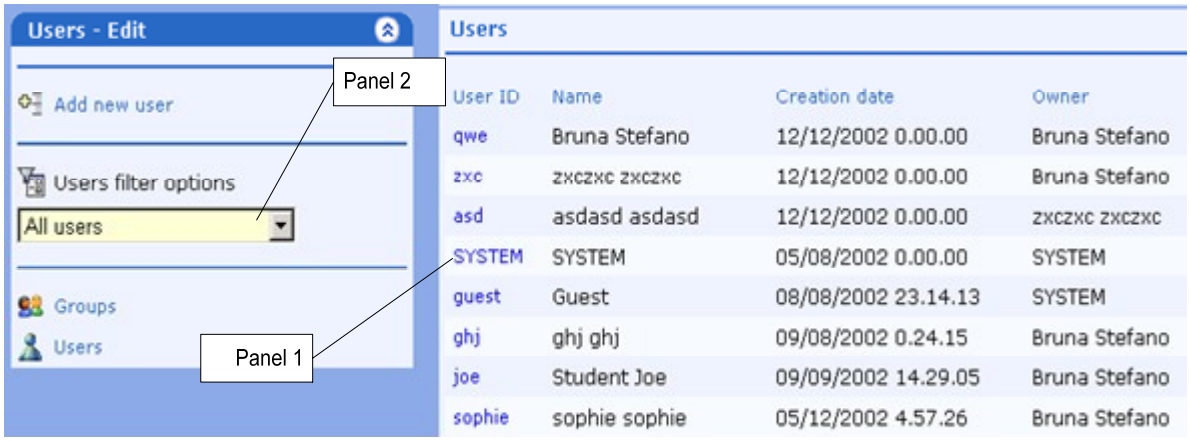


Figura 80: Pagine di gestione degli utenti.

Premendo sul link “Add new user” si ottiene una form riempiendo la quale si può inserire un nuovo utente nel sistema. L’utente che esegue tale operazione ne diviene automaticamente proprietario. Non è necessario compilare tutti i campi della form. Se un campo necessario non viene correttamente compilato o se il formato dei dati è scorretto viene visualizzato un messaggio indicante la lista della scorrettezze Un asterisco rosso compare vicino al campo non correttamente compilato.

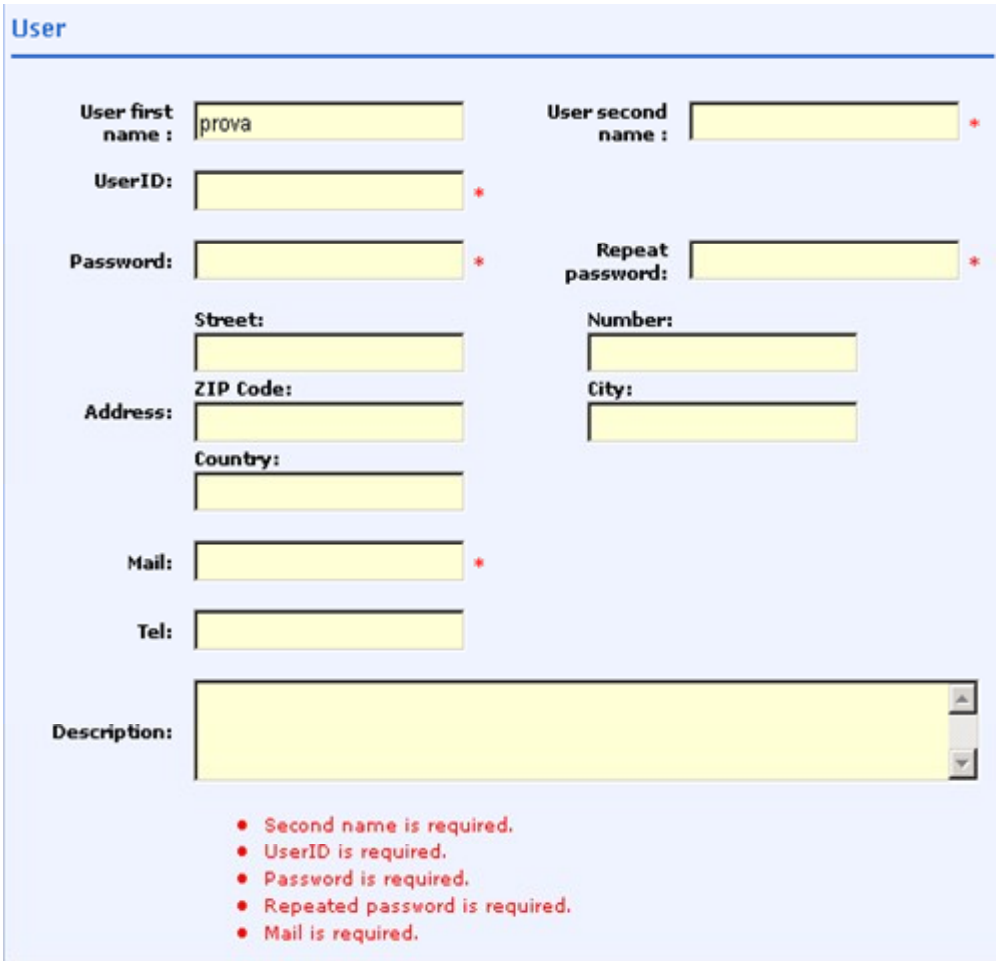


Figura 81: Form per l'inserimento di un nuovo utente. L'elenco puntato in rosso indica le irregolarità nella compilazione dei dati.

Selezionando un unente si visualizzano (Vedi Figura 82): le azioni eseguibili (Panel 1), le informazioni dell'utente con il link "edit user" per eseguire delle modifiche (Panel 2) e la lista dei gruppi a cui l'utente appartiene (Panel 3).

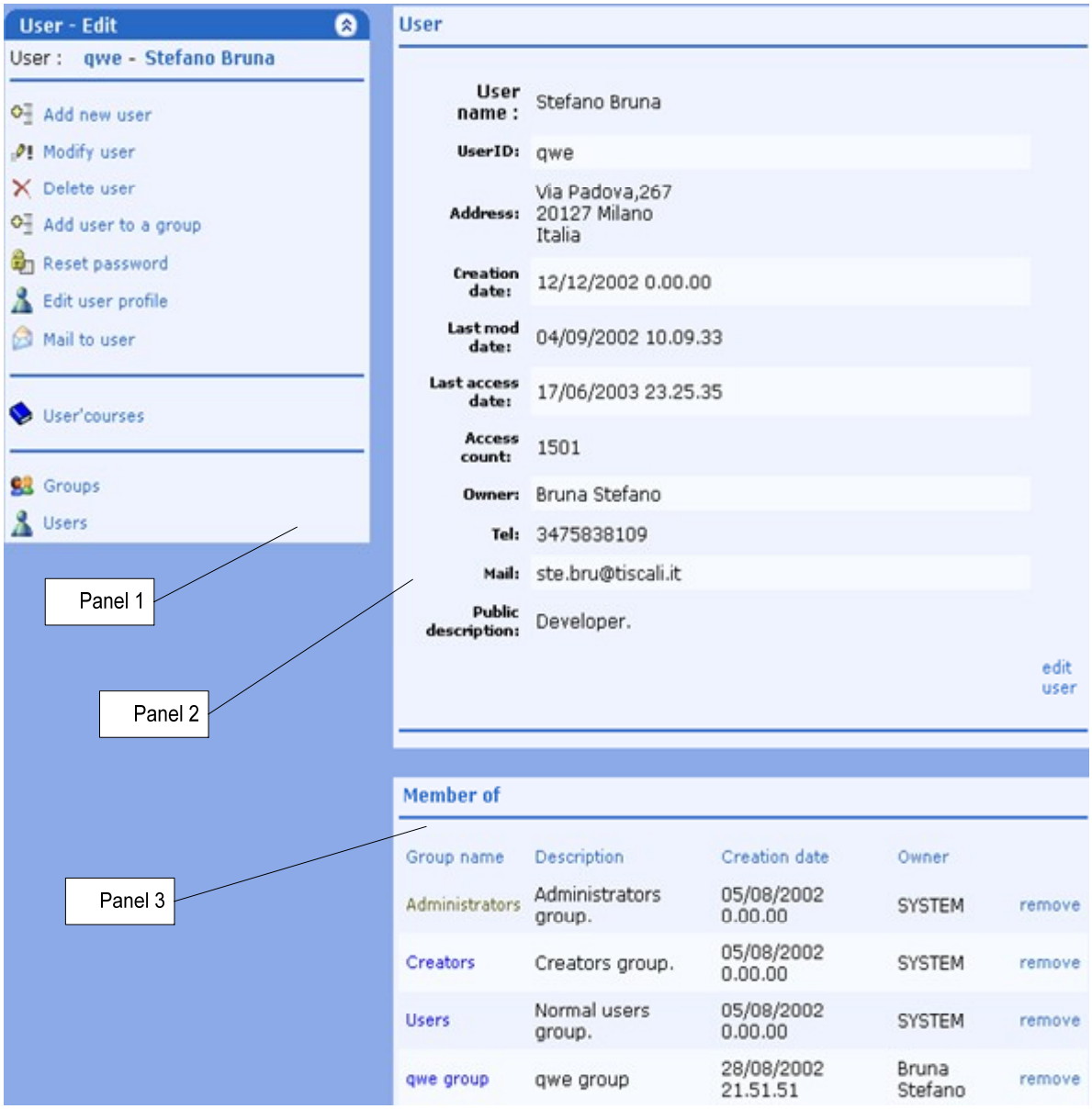


Figura 82: Pagina dell'utente.

News

Le gestione delle news permette di creare dei brevi annunci tipo bacheca in modo da avvisare gli utenti di qualche particolare evento tipo, ad esempio, la pubblicazione di un nuovo corso. Le notizie pubblicate saranno visibili da tutti gli utenti nella pagina di start di Figura 71. per accedere alla gestione delle news premere sul link “news” visibile sempre in Figura 71. In Figura 83 è possibile vedere la pagina a cui si naviga. Nel Panel 1 è presente un menu a tendina per filtrare la lista delle news che è possibile vedere nel Panel 2. E’ possibile effettuare una modifica ad una news già esistente tramite il link “edit” oppure cancellare la news pubblicata tramite il link “delete”. Premendo, invece sul link “Add news” si ottiene la form per l’inserimento di una nuova news.

Figura 83: Pagina di gestione delle news.

Subscriptions

La gestione delle sottoscrizioni permette di ottenere la lista di utenti che hanno un qualche tipo di diritto (lettura e/o scrittura e/o esecuzione) sul corso selezionato mediante il menu a tendina del Panel 1 di Figura 84. Spuntando l'opzione "Only my users" è possibile filtrare la lista di utenti relativa ad un corso.

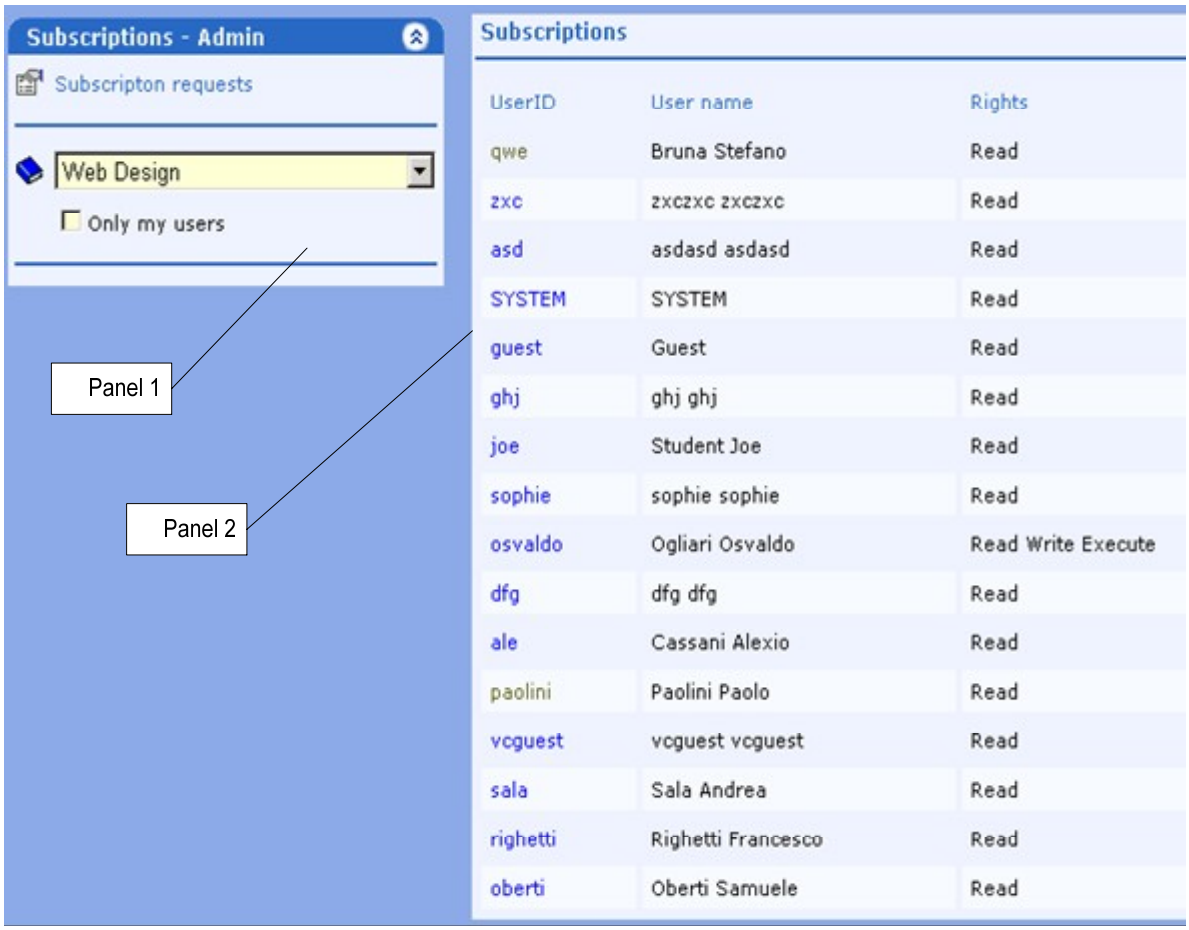


Figura 84: Gestione delle sottoscrizioni.

Premendo su link "Subscription requests" si naviga verso la pagine che visualizza la lista di richieste di sottoscrizioni ad un corso pendenti. Il concetto di sottoscrizione è identificato da un qualche tipo di indicatore che se attivo segnala che un utente è iscritto ad un corso

bensì dai diritti che l'utente ha effettivamente sul corso. Per questo motivo la procedura di sottoscrizione consta nella distribuzione di diritti per gli utenti hanno fatto richiesta. Per agevolare quest'operazione le gestione delle sottoscrizioni permette, dopo aver visto le richieste pendenti, di seguire una procedura guidata per la distribuzioni dei diritti.

- Viene visualizzata la lista di propri corsi per cui è presente una richiesta. Selezionare un corso e premere il pulsante "Next".
- Viene visualizzata la lista di utenti che hanno effettuato una richiesta di sottoscrizione per il corso precedentemente selezionato. Selezionare gli utenti. Premere il pulsante "Next".
- Scegliere se porre gli utenti in un gruppo precedentemente creato e poi distribuire i diritti al gruppo oppure associare direttamente gli utenti al corso impostando per tutti gli stessi diritti.
- Viene visualizzato un riassunto delle operazioni che saranno eseguite.

E' possibile inoltre cancellare una richiesta di sottoscrizione premendo il link "delete" corrispondente all'utente.

Projects

La gestione dei progetti permette di creare dei corsi. Per accedere a questa sezione premere sul link "Projects" nella barra di navigazione (in alto a destra) di Figura 71. La pagina che viene visualizzata presente un layout analogo alla pagina dei corsi. Nel frame a sinistra è presente una lista di progetti. Tale lista è filtrabile mediante il menu a tendina corrispondente che permette di avere la lista di tutti i progetti o solo dei propri.
Nel frame centrale è presente invece la pagina del progetto selezionato. Vedi Figura 85.
Nel Panel 1 è presente il solito menu contestuale che fornisce le azioni che è possibile eseguire sul progetto selezionato. In particolare sono presenti le seguenti voci:

- **Open Project**: per aprire un progetto e permette di crearlo o modificarlo.
- **Save Project**: per salvare il lavoro in modo che le modifiche effettuate divengano persistenti.
- **Close Project**: per terminare il lavoro e permettere, eventualmente, ad un altro utente di usare il progetto.
- **Edit Project**: dopo che il progetto è stato aperto è possibile entrare nella modalità di modifica (editing).

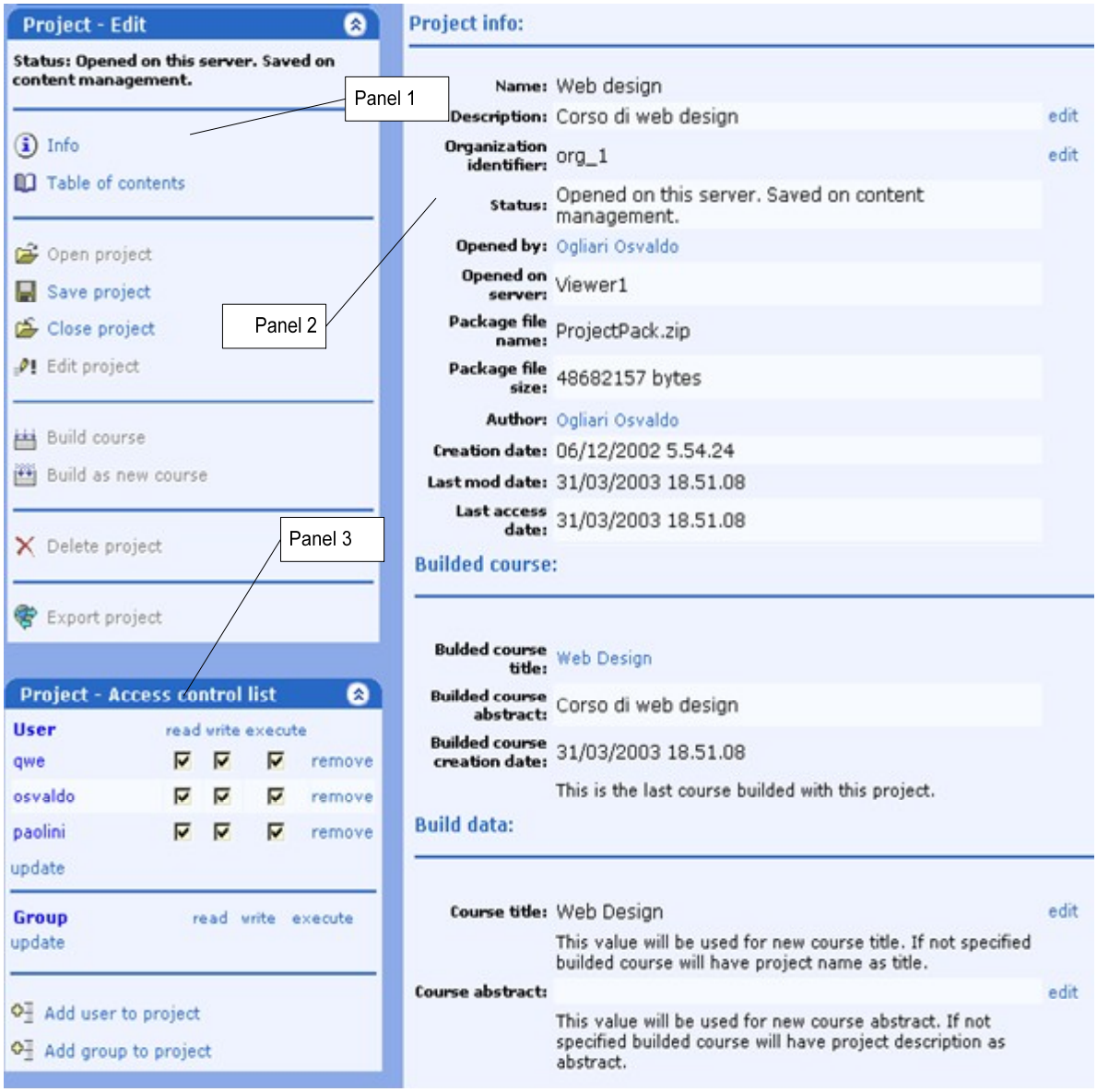


Figura 85: Pagine del progetto mediante il quale é stato creatoli corso di Web design.

Altri comandi importanti sono “Build Course” e “Build as new Course” che permettono rispettivamente di utilizzare il progetto corrente come sorgente per la creazione di un corso sovrascrivendo quello vecchio oppure di un nuovo corso. Per titolo e l’abstract del corso creato mediante questi comandi sono utilizzate le impostazioni nella sezione “Build data” del Panel 2 di Figura 85. Nella sezione “Builded course” dello stesso Panel è presente un riferimento al corso creato grazie all’ultima operazione di build . Le altre voci presenti nel Panel 2 sono informazioni relative al progetto.
Nel Panel 3 è presente la lista delle dei permessi. Per aggiungere utenti o gruppi utilizzare i comandi “Add user to project” oppure “Add group to project”. Gli utenti o i gruppi scelti dalla lista visualizzata saranno elencati nella lista dei permessi per il progetto. Per modificare i diritti spuntare le caselle corrispondenti e premere il link “update” per confermare le modifiche. Per eliminare una voce sia di un utente che di un gruppo premere il link “remove”. Tale comando non cancellerà l’utente o il gruppo ma eliminerà l’associazione fra il progetto e l’utente o il gruppo e i permessi impostati.
Accedendo alla modalità di editing si ottiene la pagina visibile in Figura 86.

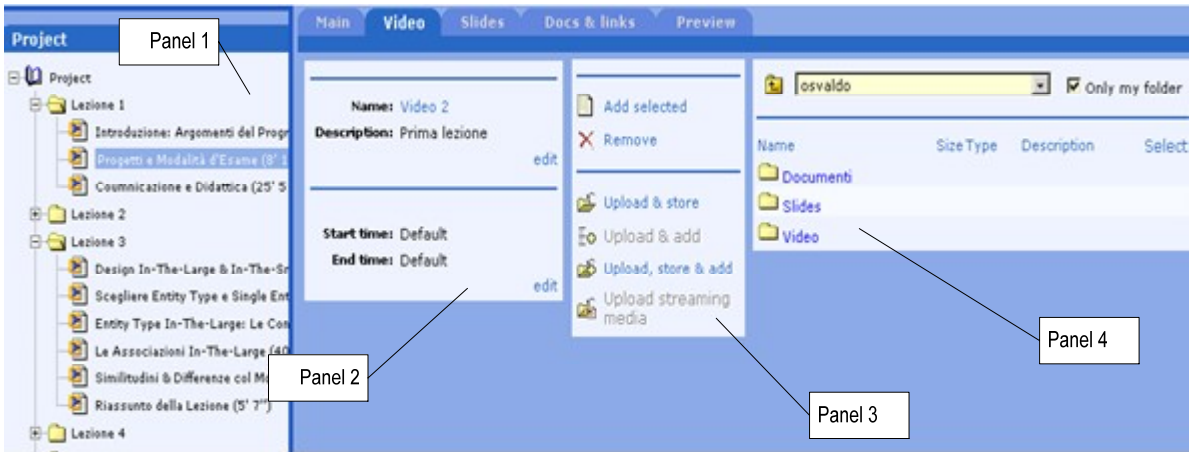


Figura 86: Modalità di editing del progetto.

Nel Panel 1 di Figura 86 é visibile la struttura ad albero delle tabella dei contenuti del corso che si sta creando. A questa struttura può essere modificata aggiungendo tre differenti tipi di elementi:

- Cartella: permette di organizzare i contenuti. Può contenere altri elementi.
- Pagina di testo formattato: permette di inserire delle pagine di testo formattato nel corso. Può contenere altri elementi.
- Contenuto multimediale: permette di inserire una pagina organizzata in quattro parti. Una contenente un video e audio, una contente delle diapositive sincronizzate con il contenuto, una contenete la lista dei segna libro presenti nel filmato e una contenente dei link e/o documenti relativi al contenuto. Può contenere altri elementi.

Nella Figura 86 è visibile la modalità di editing di un contento multimediale. I tab in alto “Main” , “Video”, “Slides”, “Docs & Links” e “Preview” permettono di passare alle varie fasi di creazione del contenuto. In particolare “Main” permette visualizza delle informazioni relative all’elemento e permette di creare dei sotto-elementi. “Video” permette di selezionare un video fra quelli presenti nel proprio archivio. Vedi Sezione “Resources”. “Slides” permette di inserire delle diapositive e tramite un numero sincronizzarle con il filmato.
E’ da notare che per ottenere una sincronizzazione è necessario che il video sia stato correttamente indicizzato e quindi siano stati inseriti nel video degli eventi “slide” con parametro indicante il numero di slide che si desidera associare all’evento. Tale indicizzazione permette anche di inserire i segnalibro nel filmato permettendo di spostarsi velocemente nell’arco temporale di esecuzione ai vari argomenti. Per effettuare queste operazioni si consiglia di utilizzare il Windows Media Indexer fornito con il resource Kit di Windows Media. Nell’ultimo tab “Preview” è possibile vedere il funzionamento del lavoro svolto senza la necessità di creare un corso e pubblicarlo solo per eseguire una verifica.
Il Panel 3 di Figura 86 mette a disposizione i comandi per selezionare le risorse da includere e permette, inoltre, di eseguire un caricamento di risorse dalla propria macchina nel caso in cui una risorsa non sia stata precedentemente archiviate. Nel Panel 4 è presente uno strumento che permette la navigazione e la selezione, anche multipla, delle risorse. Il

funzionamento di questo strumento è simile a quello di un comune file manager grazie al quale è possibile navigare all'interno della struttura ad albero dell'archivio delle risorse del proprio utente e, avendo i diritti, anche di altri utenti. È possibile, come si vedrà in seguito, condividere il proprio archivio di risorse con gli altri utenti della piattaforma.

Resources

Il sistema mette a disposizione dell'utente creatore di corsi un archivio di risorse riutilizzabile. E' inoltre possibile, impostando correttamente i permessi, condividere il proprio materiale didattico. E' possibile organizzare i files trasferiti dalla propria macchina locale al sistema i cartelle aventi la struttura tipica di un file system. E' infatti possibile vedere in Figura 87 il Panel 1 che contiene una vista della struttura ad albero che l'utente con user name "qwe" ha precedentemente creato. Mediante il menu a tendina presente nello stesso Panel è possibile vedere la struttura di cartelle solo dalla propria oppure anche le cartelle degli altri utenti. Per potervi accedere sono comunque necessari almeno di diritti di lettura. Tali diritti sono impostabili mediante il Panel 3 utilizzando la stessa logica indicata per la distribuzione dei diritti per i progetti. Nel Panel 2 sono presenti una serie di informazioni relative all'oggetto selezionato e la solita lista contestuale di operazioni eseguibili con tale oggetto. In particolare si segnalano i comandi:

- Upload resource: permette di effettuare il caricamento nel proprio archivio di una risorsa.
- Upload streaming media: permette di effettuare il caricamento di un media di grandi dimensioni del quale, durante la fruizione, verrà eseguito lo streaming. Le risorse caricate mediante questa modalità saranno riconoscibili dalla loro icona che presenterà un piccola freccia indicativa, appunto, del fatto che si tratta di un link ad una risorsa in streaming.

Selezionando un oggetto dalla lista presente nel Panel 4 si ottiene la pagina di Figura 88. Nel l Panel 1 è presente la lista di operazioni eseguibile sull'oggetto le cui informazioni sono visualizzate nel Panel 2. Mediante il link "edit" è possibile modificare il nome dell'oggetto e la sua descrizione e il suo nome. Mediante il comando "Cut resource" è possibile spostare la risorsa in un'altra cartella. Per effettuare questa operazione é necessario dopo aver premuto "Cut" nella cartella di destinazione e premere il link "Paste to folder" del Panel 2 di Figura 87. E' possibile eseguire lo stesso tipo di operazione anche sulle cartelle.

Per quanto riguarda i permessi sono valide le seguenti regole:

- Per poter accedere ad una cartella è necessario avere i diritti di lettura sulla medesima.
- Per poter modificare il contenuto di una cartella è necessario avere i diritti di scrittura sulla medesima.
- I file delle risorse presenti nelle cartelle ereditano i diritti della cartella di appartenenza.
- Per poter utilizzare un file delle risorse durante la fase di editing di un progetto di corso è necessario avere i diritti di esecuzione sulla cartella che contiene il file.

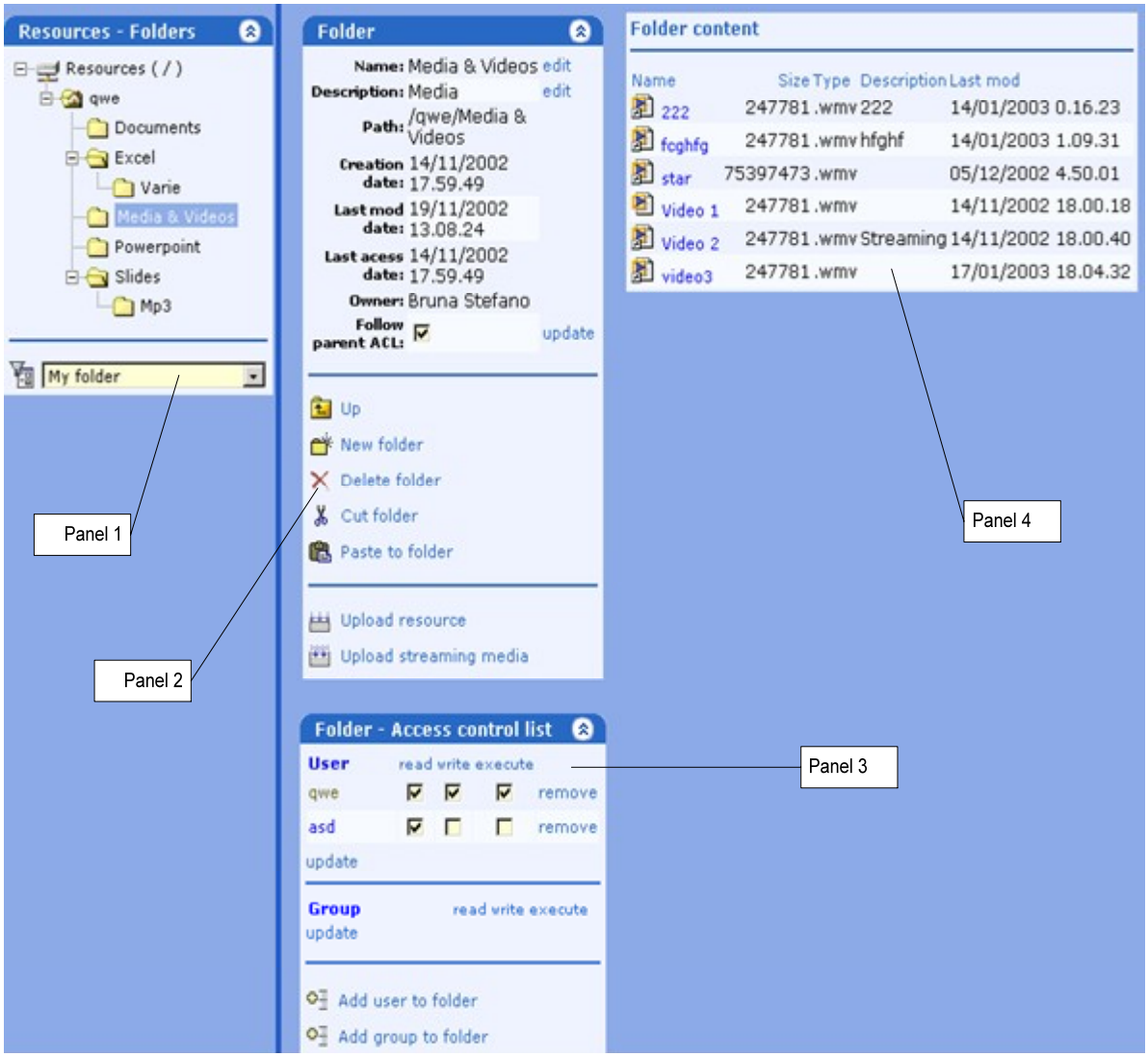


Figura 87: Pagina per la gestione della propria cartella delle risorse.

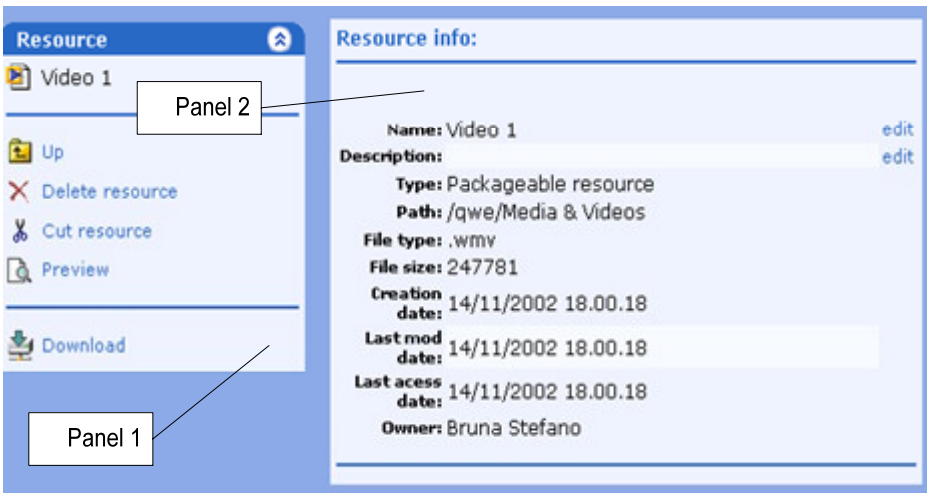


Figura 88: Pagina di un oggetto dell'archivio delle risorse.

Corso (estensione)

Avendo i diritti di Creator è possibile accedere ad una nuova sezione raggiungibile premendo il link “Course properties” della pagina del corso ottenendo la pagina di Figura 89. Sono presenti sempre la lista di comandi effettuabili con il corso, un insieme di informazioni con la possibilità eventualmente di modificarle e il pannello per la distribuzione dei diritti analogo a quello per la distribuzione dei diritti per i progetti e per le cartelle dell’archivio delle risorse.

Figura 89: Pagina delle proprietà di un corso.

In particolare si segnalano i comandi:

- Deploy course: dopo aver creato un corso tramite l’operazione di build di un progetto è necessario portarne le risorse su un server per abilitato per la fruizione. Tuttavia prima di eseguire una pubblicazione ufficiale si può eseguire questa operazione per verificare il corso.
- Publish course: dopo aver eseguito l’operazione di deployment è possibile pubblicare il corso rendendolo disponibile agli utenti aventi almeno i diritti di lettura.
- Undeploy course e Unpublish course: eseguono le operazioni inverse a quelle precedentemente descritte.

Nel pannello delle informazioni del corso sono presenti tre modalità di esecuzione selezionabili anche contemporaneamente:

- Browsable: navigazione nel corso senza tracciare le prestazioni dell'utente.
- Launchable: fruizione del corso con tracciamento e memorizzazione dello stato delle lezioni.
- Reviewable: revisione del corso con possibilità di vedere lo stato raggiunto durante l'ultima esecuzione in modalità "Launch".

La selezione di queste voci permetterà l'attivazione o disattivazione delle modalità di esecuzione che un utente può utilizzare. Vedi Panel 3 di Figura 75.

Appendice C

Manuale utente amministratore

Installazione applicazione

L'installazione dell'applicazione consta di una sequenza di operazioni non strettamente dipendenti fra loro. Di seguito viene proposta una procedure di installazione per una configurazione semplice. Per tipi di installazioni piú complesse i procedimenti sono del tutto analoghi. Si suppone che l'amministratore che esegue l'installazione abbia una certa dimestichezza con la piattaforma su cui lezi.NET funziona.

- Installazione del database.

Creare sul disco una directory per ospitare i file del database.
Copiare in tale directory il file dei dati e il file del log delle transazioni.
Utilizzando il tool enterprise manager fornito con Microsoft SQL server eseguire l'attach del dei database.
Creare un nuovo utente sempre utilizzando l'enterprise manager associandolo all'utente di sistema locale ASP.NET.
Per semplicità dare a tale utente i diritti di db owner per il database appena collegato. In questo modo le pagine aspx che impersonando l'utente APS.NET della macchina avranno la possibilità di scrivere sul database. Per una installazione più sicura è possibile definire con maggiore granularità il livello di permessi tabella per tabella. Si consiglia, comunque, di eseguire tale operazione solo dopo aver verificato il normale funzionamento di tutto il sistema.

- Installazione de lezi.NET Content Management System.

Trattandosi di una Web application è necessario, utilizzando il tool di amministrazione del Web server IIS una nuova virtual directory con nome "leziNETCMwsvc". Copiare il files della applicazione in una nuova directory su file system e farvi puntare la virtual directory precedentemente creata . Modificare il file di configurazione Web.config della Web application per poter accedere correttamente al database.

```
<appSettings>
              <add key="logFileURI" value="C:\Temp\leziNETCMSLog.txt"></add>
              <add key="excFileURI" value="C:\Temp\leziNETCMSExc.txt"></add>
              <add key="logSwitch" value="true"></add>
              <add key="dbConnString"
       value="server=(local);Trusted_Connection=yes;database=leziNETDB"></add>
</appSettings>
```

Nelle impostazioni precedenti è possibile vedere i percorsi dei file di log dell'applicazione lezi.NETCMSwsvc. Tali file sono obbligatori e devono essere creati inizialmente vuoti. L'ultima impostazione permette di impostare la connection string per il database.

Per quanto riguarda le impostazione di accesso di IIS lasciare, almeno inizialmente, la possibilità di accesso anonimo delegando le gestione della sicurezza alla modalità ‘Windows di ASP.NET.

Per quanto riguarda i permessi del file system seguire le indicazioni della successiva tabella:

Nome directory	Permessi	Utenti
leziNETCMwsvc	Allow: lettura	Everyone
leziNETCMwsvc/Course Packages	+ Allow: Scrittura	ASP_wp_account
leziNETCMwsvc/ProjectPackages	+ Allow: Scrittura	ASP_wp_account
ASP_wp_account/UserResources	+ Allow: Scrittura	ASP_wp_account

- Installazione della Web application lezi.NET

Mediante una procedura analoga a quella seguita per installazione del lezi.NET Content Management System procedere all’installazione della Web application di front end. Come nome della virtual directory utilizzare ‘leziNET’.
Modificare il file di configurazione in base alla propria configurazione:

```
<appSettings>
             <add key="logFileURI" value="C:\Temp\leziNETLog.txt"></add>
             <add key="excFileURI" value="C:\Temp\leziNETExc.txt"></add>
             <add key="log" value="true"></add>
             <add key="logSCO" value="false"></add>
             <add key="graphicsPath" value="./Graphics/"></add>
             <add key="IDServer" value="Viewer1"></add>
             <add key="leziNETCMsvcApplicationIP" value="131.175.167.20"></add>
             <add key="leziNETApplicationIP" value="131.175.167.20"></add>
             <add key="debug" value="false"></add>
</appSettings>
```

Nelle impostazioni precedenti è possibile vedere i percorsi dei file di log dell’applicazione lezi.NET. Tali file sono obbligatori e devono essere creati inizialmente vuoti.
Per i permessi del file system seguire la indicazione della successiva tabella:

Nome directory	Permessi	Utenti
leziNET	Allow: Lettura	Everyone
	Deny: Scrittura	Web Application e Anonimous users
leziNET /Wmms	+ Allow: Scrittura	ASP_wp_acount
	+ Allow: Lettura	Windows Media Services
leziNET /PubCourses	+ Allow: Scrittura	ASP_wp_account
leziNET/leziNETEditor/ProjectsWD	+ Allow: Scrittura	ASP_wp_account
leziNET/tmp	+ Allow: Scrittura	ASP_wp_account

- Configurazione dello streaming server Windows Media Services

Utilizzando l’apposito tool di configurazione impostare un nuovo punto di pubblicazione cono nome “leziNET” che punti alla directory “wmss” dell’applicazione lezi.NET. Si ricorda che l’utente che impersona lo streaming server deve avere diritti di lettura in tale directory.