POLITECNICO DI MILANO
Facoltà di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

PocketLezi:
Estensione alla multicanalità dell'ambiente Lezi.NET

Relatore: Chiar.mo prof. Licia Sbattella
Correlatore: Chiar.mo prof. Luca Mainetti

Tesina di Laurea di:
Agosti Luigi mat. 639241
Allini Alessandro mat. 638532

Anno Accademico 2003-2004

A Noi

Indice

1 Introduzione

L'applicazione PocketLezi nasce dal desiderio di espandere l'applicazione web già esistente Lezi.NET[1]. Il progetto Lezi.NET ha portato alla realizzazione di un sistema di erogazione della didattica, o *Learning Management System* (*LMS*), che soddisfa i requisiti descritti brevemente di seguito:

- **Ambiente di fruizione *Web-based***: l'utente tramite il proprio browser può fruire i contenuti di un corso in modo semplice ed automatico mediante un'interfaccia coerente ed usabile.
- **Ambiente di gestione dei corsi *Web-based***: gli utenti della piattaforma, in base ai loro diritti, hanno 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 possono 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***: è 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, è disponibile una procedura guidata che faciliti tale compito.
- **Ambiente di gestione degli utenti e dei gruppi *Web-based***: è disponibile uno strumento che permette di gestire utenti e gruppi di utenti per poter supportare processi di autenticazione e autorizzazione.

Come si può capire dal nome PocketLezi, l'obiettivo primario è quello di realizzare un ambiente web di fruizione per l'e-Learning accessibile da un dispositivo mobile, quale il PDA (o PocketPC, così chiamato per il Sistema Operativo da esso adottato). Con questo non ci si vuole limitare alla progettazione e successiva implementazione di una

[1] Il progetto Lezi.NET è stato sviluppato in ambito di tesi dall'ing. Stefano Bruna.

piattaforma completamente separata da quella originaria Lezi.NET, anzi, si vuole che i due canali di fruizione siano sincronizzati tra loro in modo tale che il sistema globale possa tenere traccia dell'apprendimento di ciascun utente indipendentemente dal device usato per il collegamento. Ciò che vogliamo verificare sono le potenzialità dei nuovi Mobile Device nell'ambito dell'e-Learning e la possibilità di espansione, in questa direzione, della piattaforma Lezi.NET.
Una seconda linea guida che è stata sempre tenuta presente, soprattutto nelle scelte tecnologiche per la realizzazione dell'applicazione, è la possibilità di espandere la stessa nell'ambito della interazione vocale; tale possibilità ha guidato le scelte inerenti la tecnologia di realizzazione di PocketLezi.

I requisiti generali del progetto sono brevemente i seguenti:

- **Ambiente di fruizione per Mobile Device**: si vuole che un qualsiasi utente, fornito ovviamente di PDA, possa connettersi via web all'applicazione e possa così fruire dei contenuti di un corso in maniera semplice e, dal punto di vista grafico, il più fedele possibile a quella dell'applicazione Lezi.NET;
- **Apertura all'interazione vocale**: prima della fase di progettazione abbiamo effettuato un lavoro di ricerca sulla tecnologie attuali per estendere l'applicazione (sia in versione PC che in versione PDA) nel campo dell'interazione vocale.
- **Adattabilità dei contenuti**: a seconda del device usato per connettersi all'applicazione, il sistema deve essere in grado di indirizzare e rendere disponibili all'utente solo quei corsi effettivamente fruibili dal device stesso;
- **Sincronizzazione dei contenuti**: il sistema deve tenere traccia del percorso di apprendimento di ciascun utente, indipendentemente dal canale utilizzato per accedere all'applicazione, in modo da permettere il passaggio da un device ad un altro, senza dover ripetere le stesse lezioni più volte.

La realizzazione di PocketLezi si inserisce all'interno del progetto MAIS [2] (Multichannel Adaptive Information System). Tale progetto ha come obiettivo lo

[2] Il team del progetto MAIS è composto da ricercatori provenienti dal Politecnico di Milano, Università Milano-Bicocca, Università di Roma "La Sapienza", Università Roma Tre, Università di Lecce, CEFRIEL, Engineering SPA, STMicroelectronics SRL.

sviluppo di modelli, metodi ed applicazioni che permettano l'implementazione di Sistemi Informativi Adattativi Multicanale, in grado di fornire servizi, tenendo conto dei diversi tipi di rete e di device di collegamento. Lo scopo di questo lavoro sarà appunto quello di fornire un esempio di applicazione Multicanale per la fruizione di contenuti didattici.

Il termine *multicanale* è generalmente utilizzato per indicare la possibilità, da parte dell'utente, di fruire di un contenuto, "in ogni luogo ed in ogni momento", potendo utilizzare diversi dispositivi. Si evidenzia come, per le soluzioni multicanale, normalmente non sono previsti meccanismi di sincronizzazione tra i diversi canali, ma si hanno *rappresentazioni* diverse di una medesima applicazione.
La multicanalità prevede la fruizione in "*modo singolo*", cioè avvalendosi di volta in volta di una sola modalità di interazione con il sistema. Nel presente documento con il termine *multicanale* si farà pertanto riferimento alla definizione proposta da IBM in [IBM PP]:
"applications designed for ubiquitous access through different channels, one channel at a time"[3].
L'obiettivo è particolarmente significativo nel momento in cui non ci si limita alla possibilità di supportare dei servizi che rispettino il minimo comune denominatore tra le caratteristiche offerte dai canali, ma quando si riesce ad offrire un'applicazione qualitativamente eccellente su un dispositivo ricco, e al contempo su un dispositivo dalle capacità ridotte si riesce a proporre una presentazione meno complessa, ma ugualmente efficace e semplice nell'utilizzo. L'applicazione deve essere portata su ogni dispositivo, sfruttandone al meglio le capacità e adattandola in modo da semplificarne l'utilizzo, in funzione della modalità scelta dall'utente per la fruizione.

Il termine *multimodale* indica "*la possibilità di interagire con un servizio attraverso differenti modalità"*. In particolare si parla di browser (o interfacce) multimodali riferendosi a quei browser che supportano differenti modalità di interazione, sia di input che di output. Le modalità cui tipicamente ci si riferisce sono, per quanto riguarda gli input, la voce, la tastiera ed i sistemi di puntamento (mouse, penna ottica, ...); le più diffuse modalità di output sono la voce, l'audio, il testo scritto e le immagini. A queste,

[3] "applicazioni create per potervi accedere da più luoghi attraverso differenti canali, un canale alla volta".

che sono esplicitamente proposte dal *World Wide Web Consortium* (W3C), se ne possono aggiungere delle altre più innovative quali, ad esempio, il riconoscimento dei gesti umani o delle espressioni del viso.
Il W3C identifica due possibili utilizzi della multimodalità: *Supplementare* e *Complementare*. L'estensione che si vuole fare dell'ambiente Lezi.NET sarà un esempio di utilizzo Complementare della multimodalità: essa visualizzerà sullo schermo di un Personal Computer una serie di opzioni e, vocalmente, richiamerà l'utente alla selezione di una di esse. Un utilizzo complementare della multimodalità può essere vantaggioso per particolari categorie di utenti, ad esempio quelli con problemi di dislessia.

Le diverse fasi del processo di sviluppo, analisi dei requisiti, progettazione, ed implementazione sono descritte nel presente documento. Il presente lavoro ha richiesto, inoltre, diverse attività di ricerca precedenti alla fase implementativa. La prima ricerca, di tipo tecnologico, è stata volta allo studio della tecnologia e della piattaforma utilizzata per l'implementazione di Lezi.NET. La seconda, invece, ha riguardato il dominio dell'applicazione, l'e-Learning, per avere una visione chiara del sistema esistente e dello standard utilizzato. Una sintesi di queste attività è presente nel capitolo 2 del documento, costituito dalle seguenti parti:

- analisi dei concetti principali e delle diverse entità coinvolte nel dominio dell' e-Learning (Cap. 2.1, 2.2);
- descrizione dei principali standard esistenti, in seguito utilizzati per la realizzazione del progetto (Cap. 2.3);
- descrizione delle principali tecnologie utilizzate durante la fase di implementazione del sistema (Cap. 2.4).

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

- *Funzionalità*: si descrivono brevemente tutte le funzionalità che devono essere soddisfatte per estendere l'applicazione Lezi.NET sia nell'ambito multicanale che multimodale (Cap. 3.1).

- *Studio di fattibilità*: in questa parte si descrive il lavoro di ricerca svolto nell'ambito dell'interazione vocale e nella sua applicabilità al caso in esame (Cap. 3.2).
- *Requisiti*: si descrivono brevemente i principali requisiti che l'applicazione realizzata deve essere in grado di soddisfare (Cap. 3.3).
- *Progettazione*: descrizione della fase di progettazione. Tale fase include la comprensione della piattaforma esistente, del mobile device da utilizzare e delle limitazioni imposte da quest'ultimo in ambito tecnologico (Cap. 4).
- *Implementazione*: la descrizione dell'attività di implementazione è volta a spiegare le soluzioni che sono state utilizzate per la realizzazione delle funzionalità richieste (Cap. 5).
- *Validazione*: descrizione dell'attività di test focalizzata principalmente agli aspetti funzionali dell'applicazione (Cap. 6).
- *Conclusioni*: considerazioni finali e possibili sviluppi futuri dell'applicazione anche in considerazione del suo utilizzo futuro per i test nell'ambito del progetto MAIS (Cap. 7).

In appendice al documento è presentato un breve manuale utente dell'applicazione.

2 Background

2.1 e- Learning

Si definisce *e-Learning* qualsiasi forma d'apprendimento che utilizzi una rete per la trasmissione, l'interazione, o l'agevolazione della didattica. La rete può essere Internet, o una Intranet aziendale. L'apprendimento può avvenire singolarmente, guidato o diretto da un computer, quello che è più conosciuto come CBT (Computer Based Training), o come parte di una classe.

Le classi *on line* si incontrano sincronicamente (allo stesso tempo) o asincronicamente (in tempi diversi), o in una qualche combinazione delle due possibilità.

Ciò che oggi è chiamato e-Learning nasce dall'integrazione di due diversi campi di sperimentazione nelle tecnologie didattiche: la *Formazione A Distanza* (FAD) e il *Computer Based Training* (CBT).

La storia della FAD si sviluppa secondo l'evoluzione delle tecnologie di comunicazione; essa parte dai corsi per corrispondenza, passa per l'emissione televisiva e arriva fino alle più recenti strutture di teleconferenza satellitare.

Il CBT, ovvero lo studio del computer come tecnologia didattica di autoistruzione, trova la sua strada soprattutto nelle discipline informatiche e nell'addestramento a determinati software e, come strumento di supporto, nella didattica delle lingue straniere.

La distribuzione di Cd-Rom in sostituzione delle tradizionali dispense cartacee può essere considerato un approccio all'integrazione di FAD e CBT, ma solo con lo sviluppo di Internet e del World Wide Web e con la diffusione del suo utilizzo, può nascere l'on-line learning, ovvero il punto d'incontro tra le due metodologie.

La capacità della rete di diffondere e distribuire informazioni, gestire dati, tracciare l'utenza, unita da un lato alle esperienze della formazione a distanza e delle sue caratteristiche emotive e cognitive e dall’altro, agli esperimenti di didattica interattiva compiuti dal CBT hanno permesso di spostare in avanti le frontiere dell'e(lectronic)-learning, dando vita a nuovi orizzonti per la didattica e la formazione aziendale.

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, quale l'interazione fra docente e studente, e al contempo di godere dei vantaggi derivanti dalla FAD, come la possibilità da parte dello studente di organizzare i tempi e, limitatamente, i modi del proprio processo d'apprendimento. In questo senso, con questo lavoro si vuole venire ancora più incontro alle esigenze dell'utente fornendolo di un'ulteriore grado di libertà: egli non sarà più vincolato alla fruizione da Personal Computer fisso ma potrà sfruttare tutti i vantaggi di mobilità derivati dall'utilizzo di un dispositivo portatile.
Con l'abbattimento delle barriere spazio-temporali, dovuto all'apprendimento elettronico, si determinano un consistente insieme di vantaggi generali, quali:

- la *facilità nell'erogazione* di formazione a distanza ad un'ampia utenza;
- non si necessita di manuali da rivedere e ristampare, bensì si ricorre a *documenti in formato digitale*, i cui contenuti sono molto più snelli da aggiornare, risultando inoltre maggiormente *riusabili* e *personalizzabili*;
- grazie a questi nuovi supporti formativi, il docente può provvedere a *cambiare "in corsa"* i materiali del corso;
- nel caso in cui l'utenza da raggiungere sia ampia e distribuita su ampi territori, i *costi dell'utilizzo* di strumenti di e-learning sono ridotti rispetto quelli relativi ad una formazione di tipo tradizionale;
- *possibilità di monitoraggio* costante e continuo degli utenti, tramite report, per controllare i vantaggi effettivi acquisiti dai partecipanti sul corso ed aiutarli a valutarne i progressi;
- *facilità d'accesso* poiché si necessita di un semplice computer (nel nostro caso è sufficiente un dispositivo palmare) e del solo accesso ad Internet;

Inoltre sono identificabili una serie di vantaggi per il singolo utente:

- *possibilità di gestire il proprio tempo* ed eliminare gli spostamenti più lunghi: è lo stesso utente a decidere come e quando frequentare il corso;

- possibilità di procedere secondo il *proprio ritmo di apprendimento*;
- l'uso di test, simulazioni e tracce interattive rende il *processo di apprendimento più semplice* e contribuisce ad una crescita professionale completa e rapida;
- possibilità di consultare in tempo reale un ampio *data base*, oltre a riferimenti successivi ad altro materiale.

Nonostante i vantaggi sopra citati, l'apprendimento elettronico non è esente da profonde critiche incentrate principalmente attorno ai seguenti punti:

- scarsa qualità della comunicazione fra studente e docente. Si ha, infatti, il dubbio che, con i mezzi che la tecnologia mette a disposizione, non si riesca a raggiungere il livello comunicativo della divulgazione verbale ottenuto *in presentia*. Tuttavia, soprattutto mediante Internet, si possono utilizzare diversi canali di comunicazione per soddisfare le più disparate necessità.
- mancanza del rapporto sociale fra studenti: tale carenza risulta evidente soprattutto nel processo di maturazione sociale della persona. Nessun sistema di apprendimento a distanza sarà mai in grado di sostituire le esperienze di vita maturate durante le lezioni in classe.

A conclusione di questa breve introduzione sull'e-Learning va ricordato come l'utilizzo della tecnologia per l'apprendimento vada comunque supportata dallo scambio umano che crea le relazioni necessarie alla condivisione della conoscenza, in modo da integrare le proprie esperienze con le esperienze altrui, assumendo nei confronti dei colleghi un ruolo di supporto e al contempo di richiesta, per la ricerca delle soluzioni adeguate in ambito formativo.
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.

2.2 Learning Management System (LMS)

Con il termine *Learning Management System* (più spesso abbreviato con la sigla LMS) si intende l'insieme di metodologie che consentono di erogare la formazione a distanza e di verificarne l'efficacia sul lungo periodo. Il Learning Management System rappresenta un complesso sistema di vasi comunicanti in grado di creare sinergie per la condivisione del sapere.

Il termine LMS può essere applicato sia a piccoli sistemi locali che permettano, ad esempio, la fruizione di un corso su CD-ROM, sia a sistemi complessi distribuiti, in cui è inclusa anche la gestione dell'utenza, processi d'autorizzazione, autenticazione, workflow, e altro ancora.

Una possibile implementazione di un LMS è visibile nella Figura 1.

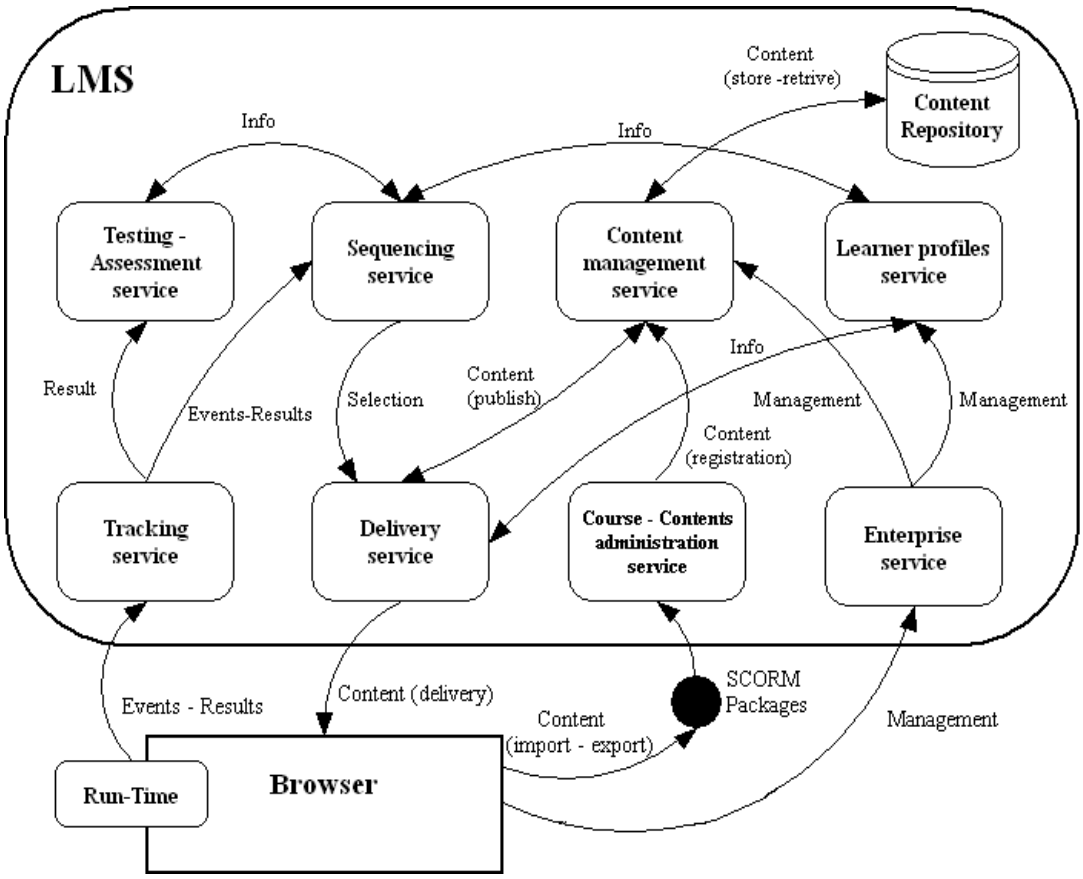


Figura 1: Componenti e servizi in una generica implementazione di LMS.

E' ora fornita una breve descrizione dei componenti rappresentati nella figura precedentemente mostrata. Dal punto di vista realizzativo, tale architettura può comunque variare considerevolmente, poiché alcune funzionalità possono essere spostate da un componente all'altro, a seconda dei dettagli che possono influenzare la fase di progettazione.

- **Run-Time**: rappresenta l'ambiente di esecuzione dei contenuti. Tale componente fornisce delle interfacce utilizzate dall'ambiente di visualizzazione, generalmente un browser, per la comunicazione con l'LMS. Ad esempio, una volta che il contenuto in esecuzione è terminato, tramite l'apposita interfaccia del Run-Time è possibile comunicare tale evento di stato in modo che l'LMS sia in grado di determinare l'azione successiva quale ad esempio la messa in esecuzione del contenuto successivo. Le interfacce messe a disposizione dal Run-Time forniscono inoltre la possibilità di ottenere e/o salvare informazioni relative allo studente e allo stato della fruizione corrente come ad esempio tempi d'esecuzione, risultato di un test, etc.
- **Delivery Service**: servizio che mette a disposizione dell'ambiente di esecuzione i contenuti. Viene generalmente implementato mediante un Web server. I contenuti provengono dal Content Management Service (o più brevemente CMS). 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 ed il resto del sistema. Fornisce l'implementazione, generalmente server-side, delle interfacce 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 del Run-Time, quest'ultimo eseguirà una chiamata, ad esempio tramite Web Service, al Tracking Service

che troverà il nome dello studente in una struttura dati [1] 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 contenuto da sottoporre all'utente. Tale servizio dovrebbe poter avere accesso alle informazioni relative all'utente, al contenuto e, sempre, alle informazioni relative all'utente nel contesto di esecuzione. In Figura 2 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.
- **Learner Profile Service**: servizio che gestisce le informazioni relative all'utente non relative ad alcun contesto di esecuzione. Ad esempio preferenze riguardanti la lingua di interazione con il sistema.
- **Enterprise Service**: servizio per la gestione di utenti, gruppi, annidamento di gruppi, permessi, access control list, autorizzazioni e autenticazione.
- **Content Repository**: repository dei contenuti. Per eseguire operazioni di inserimento di contenuti è necessario definire un formato comune[2] (fra vari LMS) per garantire l'interscambio di contenuti o aggregazione di contenti come ad esempio corsi.

[1] La struttura dati nello standard SCORM è il modello CMI (vedi Cap. 2.3.2.4)
[2] Un esempio di formato per l'interscambio di contenuti è la specifica di Content Packaging di IMS (vedi Cap. 2.3.1.2)

2.3 Standard per l'e-Learning

In questo paragrafo proponiamo una semplice introduzione agli standard che hanno guidato la realizzazione dell'applicazione Lezi.NET e, di seguito, PocketLezi.
Le specifiche IMS risultano fondamentali in quanto propongono le linee-guida che sono state implementate per descrivere ed impacchettare il materiale di apprendimento in *package* interoperabili e distribuibili.
Lo standard SCORM invece è stato utilizzato per la realizzazione dell'ambiente di esecuzione dei contenuti descritti tramite le specifiche prima menzionate. Tale standard riguarda le specifiche per l'avvio, la comunicazione ed il tracciamento dei contenuti all'interno dell'ambiente di interazione Web (ovvero la piattaforma LMS).

2.3.1 IMS (Instructional Management System)

IMS Global Learning Consortium è un'associazione internazionale composta da organizzazioni governative, commerciali e formative. Il suo ruolo consiste nello sviluppo e nella promozione di specifiche linee-guida che hanno l'obiettivo di facilitare l'erogazione dei corsi di formazione a distanza. In particolare, i requisiti IMS garantiscono un grado di interoperabilità tra applicazioni e servizi differenti: la loro adozione a livello internazionale facilita il dialogo tra diversi ambienti e piattaforme di formazione elettronica.
Le specifiche e le linee guida prodotte riguardano standard che possono essere utilizzati per la progettazione di sistemi 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), i quali 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 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 essere 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.

2.3.1.1 Specifiche IMS

L'intero framework specificato dai documenti IMS risulta essere ampio e complesso. Al fine di ridurne 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 2 fornisce una visuale completa del framework e delle relazioni che intercorrono fra le varie specifiche.

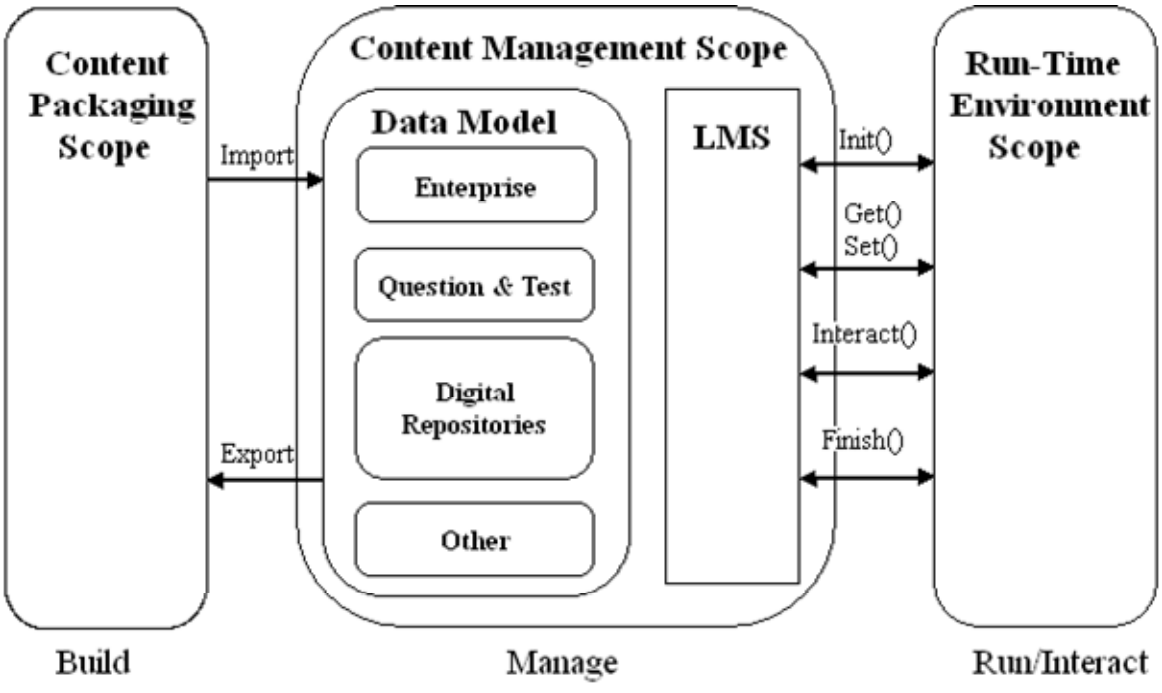


Figura 2: Relazioni fra i vari scope di specifica IMS

Ogni 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 ed 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.
Riportiamo qui di seguito una breve descrizione di tali specifiche per il cui approfondimento si rimanda al sito ufficiale IMS: http://www.imsglobal.org.

- **Metadata Specification**: definisce una struttura informativa associata ai contenuti per facilitarne la ricerca e la fruizione. In particolare specifica i campi, le definizioni, il formato dei dati, le regole ed i controlli necessari per descrivere in modo esaustivo e formale tutte le entità che si incontrano nei sistemi di e-Learning.
- **Content Packaging Specification**: facilita l'interscambio dei contenuti tra diversi LMS fornendo una specifica sull'impacchettamento dei contenuti e delle informazioni di supporto.
- **Question-Test Specification**: descrive la struttura per la gestione di semplici domande o test necessari ai corsi nei quali è importante avere un feedback della conoscenza acquisita dallo studente.
- **Enterprise Specification**: descrive delle entità che semplificano il trasferimento di informazioni organizzative degli studenti e dei gruppi attraverso diversi sistemi.
- **Digital Repository Interoperability**: provvede a fornire delle linee guida per realizzare un accesso omogeneo alle funzioni di ricerca e invio dei contenuti nei più comuni repository.

2.3.1.2 IMS Content Packaging Specification

Di tutte le specifiche IMS sopra riportate, l'applicazione PocketLezi implementa solamente le IMS Content Packaging Specification, in quanto il progetto Lezi.NET, da cui il nostro prende piede, effettua tale scelta. Sempre per lo stesso motivo, per altri

sottosistemi quali il Run-Time Environment, sono state seguite le specifiche SCORM (di cui si parlerà in seguito).

Tali specifiche forniscono le linee-guida per descrivere ed impacchettare il materiale di apprendimento, quali ad esempio un singolo corso oppure una collezione di corsi, in *package* interoperabili e distribuibili. Per garantire tale interoperabilità e trasportabilità tra i vari LMS, è necessario fornire uno standard comune per la creazione di questi corsi pacchettizzati.

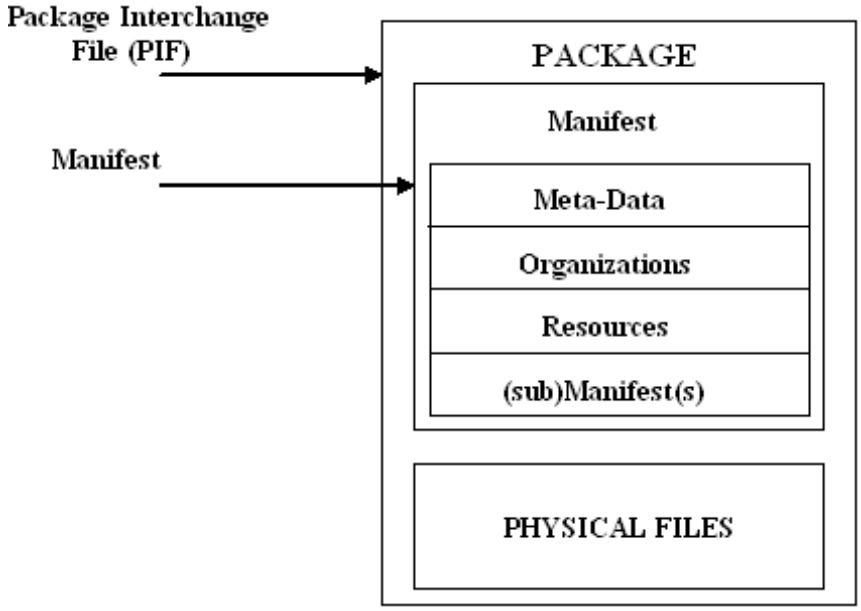


Figura 3: IMS Package

L' IMS Package, rappresentato nella figura sovrastante, è formato principalmente da due elementi: un file XML che descrive l'organizzazione e le risorse contenute nel pacchetto ed i file fisici dei contenuti descritti in tale file XML.

Il file speciale XML è denominato *IMS Manifest* perché i contenuti e l'organizzazione del corso sono descritti in termini di *manifests*; il nome di tale file dovrà essere obbligatoriamente **imsmanfest.xml**.

Per il loro utilizzo, i file sono organizzati in una singola directory che include il manifesto, mentre, per essere trasportati, vengono compressi in un singolo file (Package Interchange File).

- **Package Interchange File**: è un singolo file (per esempio, 'zip', 'jar', 'cab') che include un manifest di alto livello chiamato "imsmanifest.xml" e tutti gli altri file da esso descritti (il formato del file zip deve essere conforme alla specifica RFC1951).
- **Package**: è una directory, che include un file speciale XML, tutti i documenti di controllo necessari ad esso (quali XSD o DTD) e le risorse fisiche reali. Le risorse fisiche possono essere organizzate in sotto-directory.

- **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:
  - **Metadata section** : un elemento xml che descrive il manifesto;
  - **Organizations section**: un elemento xml che descrive zero o più organizzazioni dei contenuti;
  - **Resources section**: un elemento xml contenente i riferimenti a tutte le risorse necessarie al manifesto, i Metadata che descrivono le risorse e i riferimenti a risorse esterne;
  - **(sub)Manifest(s)**: uno o più manifesti opzionali annidati.

- **Physical Files**: sono file di testo, immagini o altre risorse descritti dal manifesto che possono essere anche organizzati in sotto-directory.

2.3.1.3 Struttura dell'IMS Manifest

Data l'importanza che tale elemento riveste all'interno del lavoro svolto, riportiamo una semplice descrizione dell'IMS Manifest. Per una descrizione formale rimandiamo come sempre al sito ufficiale IMS (http://www.imsglobal.org).

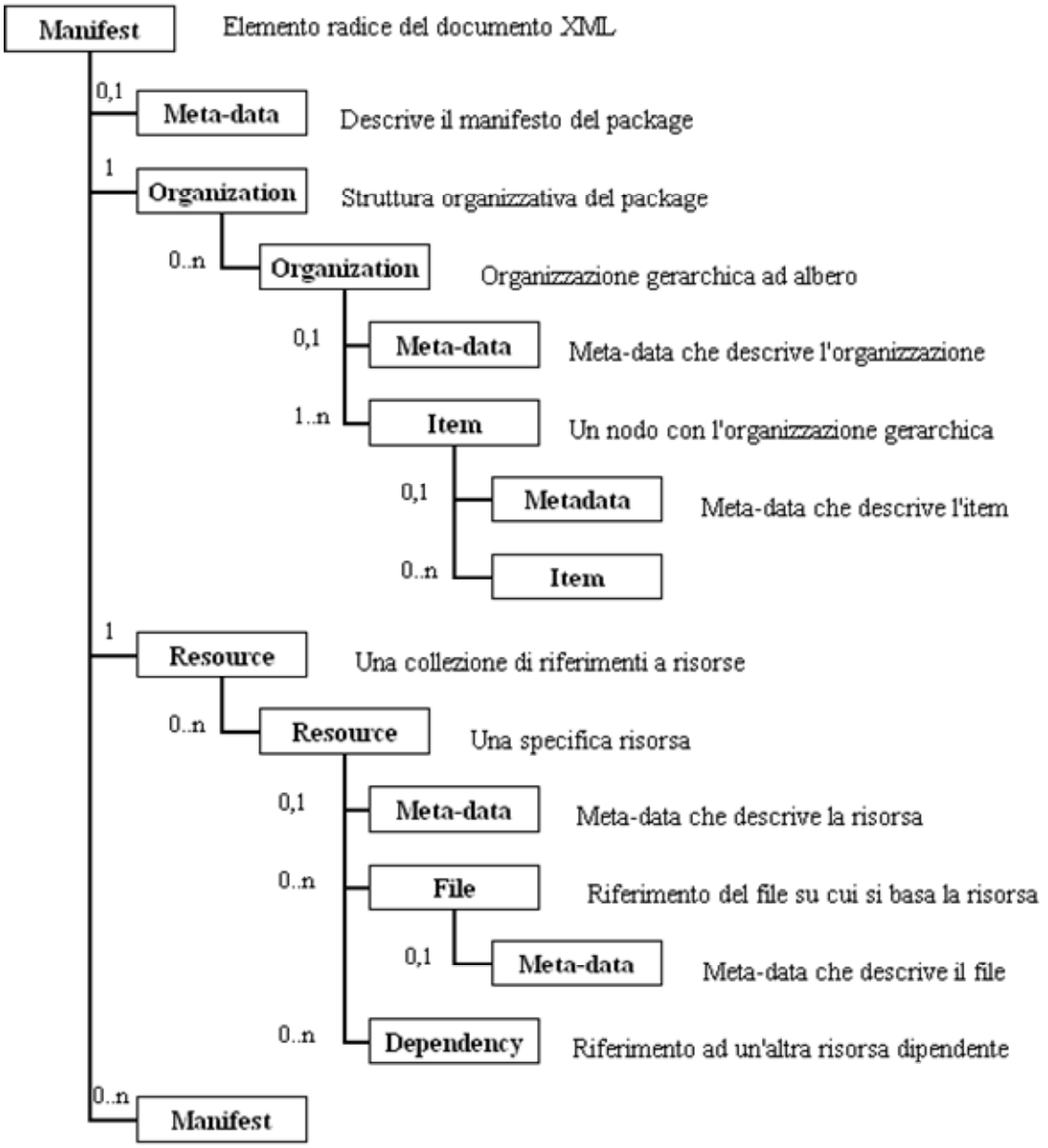


Figura 4: Struttura IMS Manifest

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.

Per chiarire ulteriormente il concetto di IMS Manifest, ed in particolare del suo elemento Organization, riportiamo ora un esempio di struttura gerarchica con cui vengono presentati i contenuti di un corso, e di come questa venga interpretata dall'LMS:

Esempio 1: (a) elemento organization che definisce la struttura gerarchica di un corso

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

(b) come un LMS interpreterà la struttura sovrastante

- Lezione 1
  - Lezione 1.1
- Lezione 2
- Lezione 3

2.3.2 ADL-SCORM

SCORM [3] (Sharable Content Object Reference Model) definisce il modello di aggregazione dei contenuti per la Formazione a Distanza (FAD) e l'ambiente di run-time per i *Learning Objects* (in futuro abbreviati in *LO*).

SCORM è un insieme di specifiche adattate da molte fonti per fornire una suite completa di strumenti di e-learning che permettono interoperabilità, accessibilità, e riusabilità dei contenuti formativi basati su web. Il lavoro portato avanti dall'*ADL Iniziative* per sviluppare lo standard SCORM è anche un processo per coordinare diversi

[3] Vedi ADL-SCORM

gruppi ed interessi. Questo modello di riferimento ha come obiettivo quello di coordinare le tecnologie emergenti con implementazioni commerciali e/o pubbliche.
Disporre di uno standard comune permetterebbe infatti il trasferimento dei contenuti da un'architettura all'altra, poterli integrare tra loro, saperli scegliere in base a caratteristiche e classificazioni univoche ed infine poterli certificare con un attestato che indichi le competenze acquisite.
L'architettura di SCORM è composta da quattro elementi essenziali:

1. *Learning Object (LO)*: la cellula minima della quale si compone un corso. Un LO è una risorsa didattica "modulare", una risorsa didattica che si può riusare senza la necessità di modificarne i componenti. Lo standard SCORM è stato elaborato per rendere generalmente riusabili i LO, sulla base dell'esplosione del World Wide Web. Con la certificazione SCORM, i LO possono essere usati in qualunque programma per il supporto alla didattica (LMS) che sia costruito secondo il modello previsto dalla standard SCORM.. Tale modello prevede essenzialmente la separazione dei contenuti didattici dall'esecuzione dei sistemi di istruzione. I contenuti didattici sono incapsulati nei LO e resi reperibili universalmente sulla rete attraverso dei Metadata;
2. *Learning Management System (LMS)*: il sistema di gestione del corso che ne consente la fruizione;[4]
3. *Course Structure Format (CSF)*: file d'interscambio in grado di tradurre lo stesso corso in LMS differenti;
4. *Runtime*: il sistema che avvia il corso, soddisfacendo le richieste dell'utente.

Le specifiche SCORM sono costituite da 3 testi, la cui struttura è la seguente:

1. **Overview**: costituisce una panoramica generale dei contenuti e della utilità dello SCORM;
2. **Content Aggregation Model**: contiene le indicazioni per identificare ed aggregare le risorse in uno strutturato contenuto di apprendimento. In esso si descrivono una nomenclatura per i contenuti, il meccanismo di "impacchettamento" (*SCORM Content Packaging*) ed i riferimenti al modello dell'IMS (IMS Learning Metadata Information Model) basato sulle specifiche

[4] Per approfondimenti si rimanda al capitolo 2.2

per i LO dell'IEEE (LOM Learning Object Metadata). Il documento è composto dalle tre seguenti sottosezioni:

i. Metadata Dictionary
ii. Content Packaging
iii. Contenti Aggregation Components

3. **Run-Time Environment**: include le linee guida per rendere possibile il collegamento all'archivio dei contenuti, la comunicazione con esso ed il reperimento di un determinato contenuto. Infatti, il Run-Time Environment è il meccanismo che rende possibile la riusabilità dei Learning Object e la interoperabilità tra molteplici Learning Management System. Più chiaramente, esso costituisce il comune meccanismo dei LO di comunicare con un LMS; per fare questo occorre che sia predisposto un vocabolario che formi la base della comunicazione.

2.3.2.1 SCORM Content Aggregation Model

Il volume 2 dello SCORM si occupa di come aggregare ed assemblare risorse educative per costruire e distribuire package, ovvero unità didattiche di varia dimensione a loro volta ricomponibili.

La nomenclatura utilizzata per definire i componenti è basata sui termini **Asset**, che identifica un componente "atomico", non ulteriormente divisibile (una pagina HTML, una parte di essa, una immagine, un'animazione, un programma) e **SCO** (*Sharable Content Object*), il vero LO costituito da uno o più Asset e da un Wrapper che consenta la comunicazione con la piattaforma di lancio dello SCO. Nella filosofia SCORM lo SCO è l'unità minima "lanciabile" e tracciabile (nel senso di poter verificare che un utente abbia iniziato e finito la fruizione di quel contenuto) da una piattaforma. Sia gli Asset che gli SCO sono descrivibili tramite Metadata, allo scopo di assicurarne la facile reperibilità ed il semplice riuso.

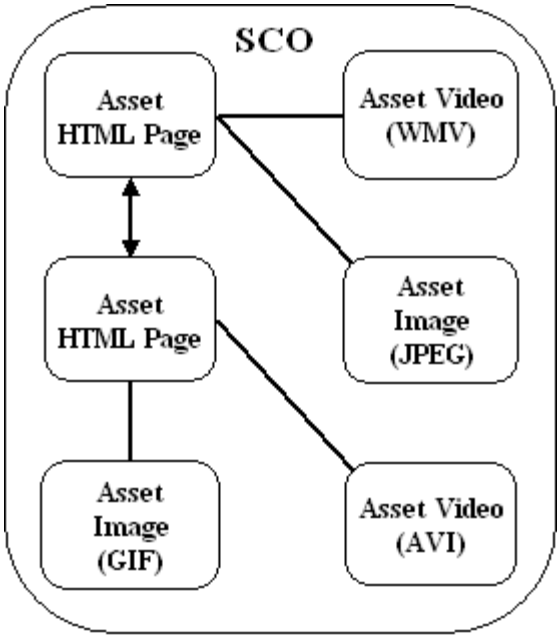


Figura 5: Esempio di SCO. All'interno dello SCO sono presenti diversi asset. La navigazione fra i diversi asset è possibile ed è responsabilità dello SCO fornire gli hyperlink necessari.

I singoli SCO sono infine collegati tra loro per formare una unità didattica (un modulo, una lezione, un intero corso). 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 2.3.1.2, 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 ed ai riferimenti ai file degli asset.
- Le **Metadata Dictionary** che definiscono dei Metadata 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 Metadata vengono rappresentati mediante elementi presenti nel manifest e documenti xml *standalone* associati a ciascuno SCO. Un riferimento a tali file è comunque presente nel manifest.

2.3.2.2 SCORM Run-Time Environment

Il Run-Time Environment riguarda le specifiche per l'avvio, la comunicazione, il tracciamento dei contenuti all'interno dell'ambiente di interazione Web (la piattaforma

LMS). Esso è il meccanismo che rende possibile la riusabilità dei LO e l'interoperabilità fra piattaforme differenti. L'ambiente di Run-Time è responsabile del tracciamento delle attività svolte dall'utente a livello di SCO (i singoli asset non sono tracciabili).
Per adempiere a queste molteplici attività il Run-Time Environment è composto da tre parti: Launch, Application Interface (API), ed un Data Model (vedi AICC-CMI) che garantiscono rispettivamente un modo comune di iniziare le risorse d'apprendimento, un comune meccanismo di comunicazione con un qualsiasi LMS ed un linguaggio predefinito che costituisca la base della comunicazione stessa. Nella figura seguente è possibile osservare le relazioni che intercorrono tra le suddette entità:

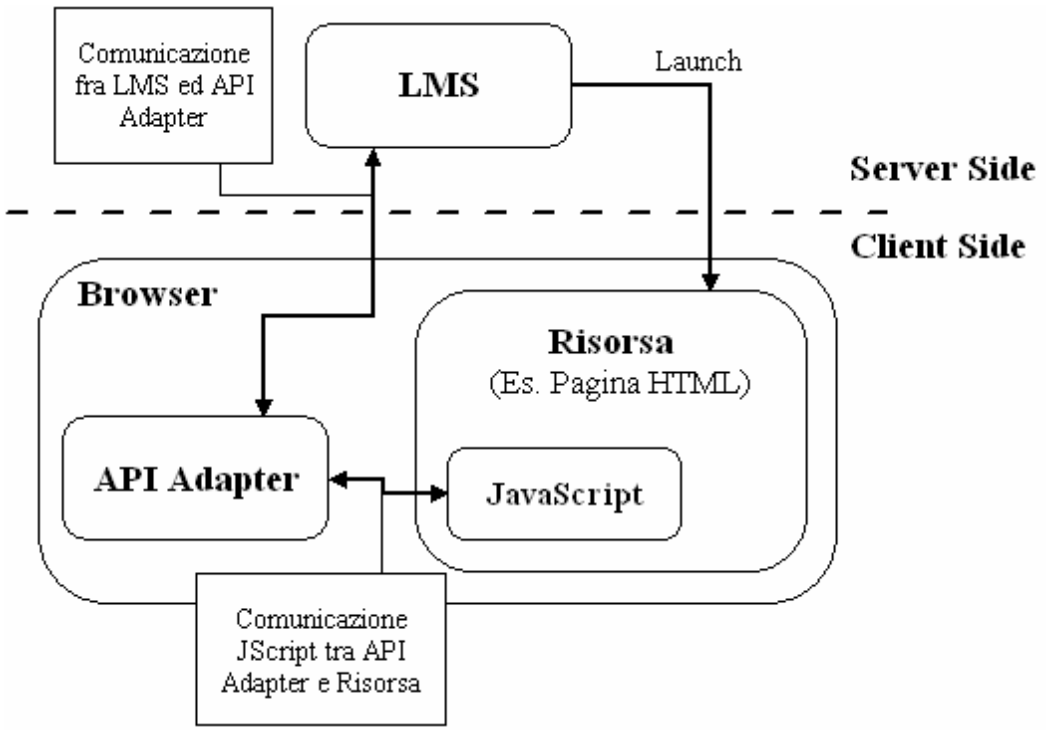


Figura 6: Interazione fra LMS e risorsa.

2.3.2.3 Interfaccia di Comunicazione (API)

L'utilizzo di un'interfaccia comune oltre a garantire interoperabilità e riusabilità delle risorse premette di semplificare il processo d'implementazione di un sistema di learning completo.
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 all'LMS di inizializzare la propria struttura dati (proprietaria) ed il Data Model per poter supportare l'esecuzione. E' obbligatorio che uno SCO esegua questa chiamata prima di qualsiasi altra.
- **LMSFinish**: lo SCO esegue questa chiamata quando ritiene che il canale di comunicazione con l'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 dall'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 dall'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*[5] indica un campo del Data Model inizializzato probabilmente dall'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.

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. Nella figura seguente è possibile vedere gli stati e le corrispondenti chiamate LMS possibili.

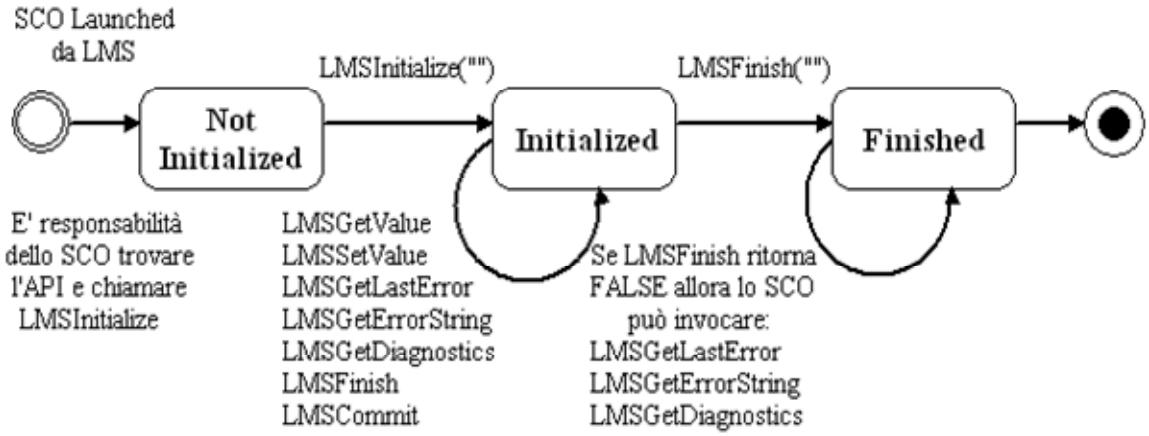


Figura 7: Transizioni di stato dello SCO.

- **NotInitialized**: è lo stato in cui si trova lo SCO fra il momento del lancio da parte dell'LMS ed il momento in cui lo SCO stesso chiama LMSInitialize. Durante questo stato lo SCO cerca nella struttura gerarchica dei frame 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 LMSInitialize e LMSFinish. In questo stato è possibile chiamare un qualsiasi metodo dell'API tranne LMSInitialize.

[5] Della struttura dati parleremo più approfonditamente nel prossimo paragrafo.

- **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 la gestione dell'errore.

2.3.2.4 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, fosse necessario registrare le prestazioni di uno studente, sarebbe 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 il 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, usare 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 dall'*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.
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 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
```

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. Per permettere agli SCO di capire quali campi siano effettivamente presenti nel modello, sono presenti dei campi speciali accessibili in sola lettura che contengono la lista dei Metadata 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.
Per una visione più dettagliata del modello si rimanda al sito ufficiale http://www.adlnet.org.

2.4 Tecnologie

2.4.1 SOAP-WSDL-UDDI

SOAP (*Simple Object Acces Protocol*) è un protocollo di comunicazione basato su una tecnologia già esistente e funzionante, in quanto utilizza per il trasporto dei messaggi di richiesta/risposta il protocollo HTTP nel quale i parametri ed i comandi sono codificati con XML; in questo modo si riesce ad eliminare il problema della differenza di piattaforma, questo perchè XML ha una tecnologia Web estremamente flessibile ed estensibile. Questo non definisce un modello di programmazione o implementazione di specifiche semantiche, ma permette di esprimerle fornendo un modello di impacchettamento modulare e meccanismi per la codifica dei dati.
Il SOAP è composto da tre parti che, nonostante vengano descritte assieme, sono funzionalmente diverse; in particolare lo sviluppo e le regole di codifica sono definiti in *namespace* diversi per ottenere maggior semplicità. Queste parti sono:

- *SOAP Envelope Construct*: definisce una struttura per esprimere cos' è un messaggio, chi lo deve trattare e se è obbligatorio o meno.
- *SOAP Encoding Rules*: definisce un meccanismo che può essere usato per scambiare tipi di dati.
- *SOAP RCP Representation*: definisce la convenzione che viene usata per procedure di chiamata remota e risposte.

Il principale obiettivo di SOAP è garantire semplicità ed estendibilità, questo significa che molte caratteristiche dei tradizionali sistemi di comunicazione non fanno parte delle specifiche SOAP. Vediamo ora, per farci un idea, due esempi di messaggi SOAP all'interno di richiesta e risposta HTTP:

Esempio 2: (a) Messaggio SOAP-Request

```
POST /StockQuote HTTP/1:1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope>
xmlns:SOAP-ENV="HTTP://schemas.xmlsoap.org/soap/envelope/"
SOAP-
ENV:encodingStyle=HTTP://schemas.xmlsoap.org/soap/encoding/>
      <SOAP-ENV:Body>
            <m:GetLastTradePrice xmlns:m="Some-URI">
            <symbol>DIS</symbol>
            <m:GetLastTradePrice>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Esempio 2: (b) Messaggio SOAP-Response

```
HTTP/1.1 200 OK
Content-Type: Text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope>
xmlns:SOAP-ENV="HTTP://schemas.xmlsoap.org/soap/envelope/"
SOAP-
ENV:encodingStyle=HTTP://schemas.xmlsoap.org/soap/encoding/>
      <SOAP-ENV:Body>
            <m:GetLastTradePriceResponse xmlns:m="Some-URI">
            <Price>1000$<Price>
            <m:GetLastTradePriceResponse>
      </SOAP-ENV:Body>
</SOAP-ENV: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 byte 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 (*Web Service Description Language*) è un formato XML che ha come obiettivo la descrizione dei Web Service esposti da un server. Le operazioni ed i messaggi sono descritti in maniera astratta e non collegati ad un preciso protocollo di comunicazione o meccanismo di trasporto.
In particolare un documento WSDL è un file XML che presenta la suo interno diverse sezioni:

- *Service*: informazioni su un servizio esposto;
- *Binding*: i protocolli utilizzabili per raggiungere il servizio;
- *PortType*: le porte di ingresso e di uscita del servizio;
- *Message*: i messaggi che si possono inviare e ricevere;
- *Types*: i tipi di dati utilizzati in ingresso ed in uscita con la descrizione della loro struttura.

La sezione *Service* descrive un singolo servizio. Poiché un documento WSDL può contenere la descrizione di più servizi, per ciascuno di essi, avremo una sezione *Service* corrispondente. Di ciascuno avremo le seguenti informazioni:

- Il nome univoco tramite l'attributo *name* del tag *Service*;
- La porta o le porte tramite le quali possiamo raggiungere il servizio, distinte per nome (attributo *name*) ed identificate da un determinato *binding* che farà riferimento ad un'altra sezione apposita del WSDL;
- La URI (*Uniform Resource Identifier*) presso la quale possiamo ottenere il servizio, tipicamente si tratterà di una URL (*Uniform Resource Locator*);
- Possiamo avere anche delle informazioni descrittive del servizio all'interno di un tag *documentation*.

Nella sezione dei *binding* troviamo, raggruppate per nome, le informazioni di binding relative ad un dato servizio. In particolare possiamo scoprire:

- Il nome (attributo *name*) del binding per poterlo associare ad un servizio;

- Il tipo di binding (attributo *type*) che punterà ad una definizione di porta (*portType*), di cui parleremo successivamente;
- La sezione *soap:binding* che è una estensione a WSDL, infatti utilizza un diverso alias di Namespace, indica che tipo di interazione avremo tra i SOAP Node. Possiamo specificare tramite l'attributo *style* il fatto che avremo nel *soap:body* dei documenti SOAP trasmessi in un dialogo RPC (*Remote Procedure Call*) oppure un dialogo basato su documenti XML. Inoltre, sempre nell'elemento soap:binding possiamo indicare con quale protocollo vogliamo far viaggiare i messaggi SOAP.
- Abbiamo quindi l'identificazione dell'operazione che viene descritta da questo binding nell'attributo *name* del tag *operation*;
- Infine abbiamo la descrizione, tramite un'ulteriore estensione a WSDL, di come debbano essere trasmessi i messaggi in input ed in output (*literal* od *encoded*) e di quale sia la *soapAction* corrispondente al binding corrente.

La sezione binding puntava a quella *portType*; in questa troviamo dei puntatori alla struttura dei messaggi di ingresso e di uscita associati ad una certa porta, identificata da un nome. Possiamo avere anche delle informazioni sulla porta all'interno di un tag *documentation*.

Si noti il fatto che il nome dei due messaggi di input e di output è associato, tramite un alias, al Namespace corrispondente al servizio.

La sezione *message* include:

- Innanzitutto il nome dei messaggi, che viene utilizzato per identificarli e riconoscerli, per esempio nel momento in cui sono referenziati nella sezione portType;
- Quindi abbiamo per ciascun messaggio l'indicazione dell'elemento che, descritto nella sezione *types*, rappresenta la struttura del *soap:body* da fornire.

Vediamo quindi l'ultima sezione descrivente le tipologie di dati che possono essere scambiate all'interno dei messaggi SOAP. La sezione *types* si appoggia ad *XML Schema Definition* (XSD) per la descrizione dei tipi. Questa non è una scelta obbligatoria. La specifica che descrive WSDL richiede l'utilizzo di una qualsiasi grammatica in grado di descrivere tipi di dati, consigliando per ora di utilizzare *XSD*.

Per dettagli inerenti lo standard WSDL consultare [W3C.WSDL].

Per poter avere un indice organizzato dei Web Service disponibili in un ambito non solo locale è possibile utilizzare un sistema di directory per i Web Service, chiamato UDDI (*Universal Description, Discovery and Integration*), 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].

2.4.2 Microsoft Framework .NET

Con *Microsoft .NET Framework*[6] si indica la piattaforma di elaborazione per lo sviluppo e la distribuzione di applicazioni in ambiente Windows e tutte le tecnologie coinvolte, come ad esempio: ASP.NET, C#, VisualBasic.NET, JScript.NET.
Il Framework .NET consiste di due parti principali: il *CLR* ed una libreria di classi (*.NET Framework Class Library*) che include ad esempio ASP.NET ed un ambiente per lo sviluppo di applicazioni Client Windows Forms. Entrambi si basano sulle *Framework .NET Base Classes*.

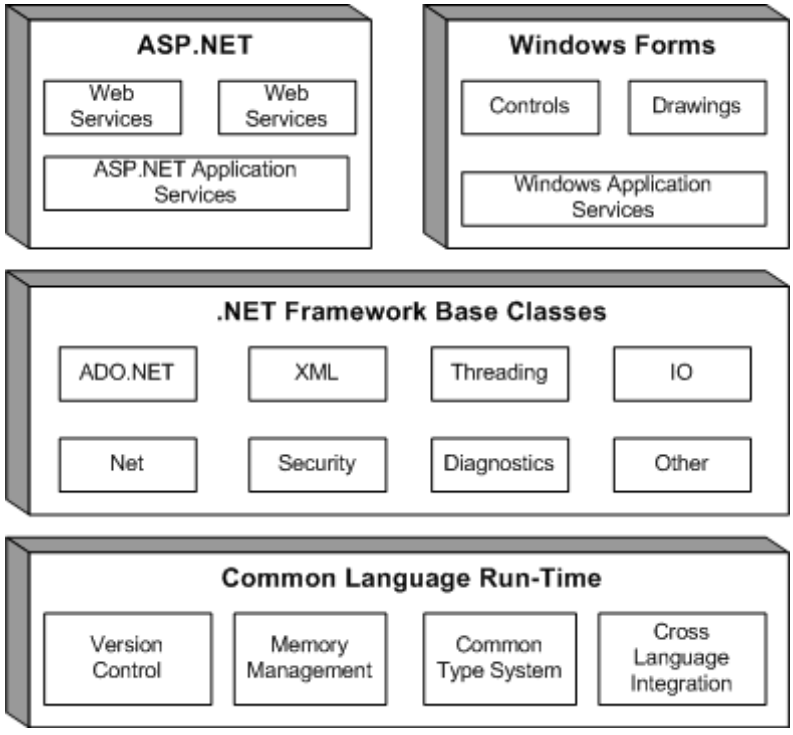


Figura 8: Componenti del Framework .NET

[6] Il Microsoft .NET Framework è una possibile implementazione ed estensione della CLI (standard ECMA-334 e 335). Infatti esistono già altre implementazioni del CLI quali: SSCLI, Mono (per Linux), DotGNU e IntelOCL.

2.4.2.1 CLR (Common Language Runtime)

Il CLR rappresenta il cuore del Framework .NET, in quanto fornisce un ambiente nel quale l'applicazione può essere eseguita. Come suggerito dal nome, il CLR è concepito per supportare molti linguaggi di programmazione.
Il compilatore, partendo dai sorgenti, crea un eseguibile contenente un linguaggio intermedio detto IL (Intermediate Language) o MSIL.

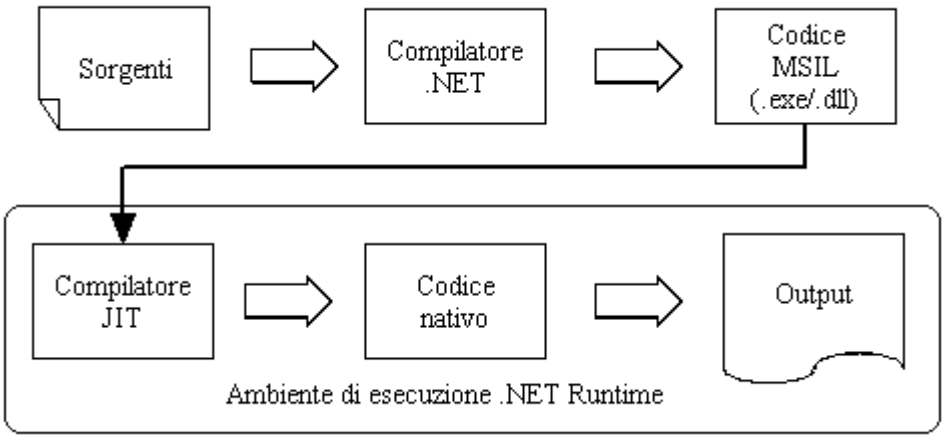


Figura 9: Processo di compilazione del Framework .NET

Il CLR grazie al *JIT Compiler* (*Just in time*) compila il codice dell'applicazione trasformandolo in codice macchina nativo, questa conversione viene realizzata in fase di esecuzione dell'applicazione e riguarda solo quelle parti di codice che vengono richieste dalle varie funzioni del programma. In questo modo si elimina la fase di *overhead* dovuta alla compilazione in codice nativo di un intero programma.
Nella logica .NET, il software è costituito da diversi assembly, con funzionalità di librerie, e da un file assembly principale (avente estensione exe), necessario per l'avvio del programma stesso. Ogni assembly è autodescrittivo, nel senso che al suo interno è presente il codice IL con dei Metadata che forniscono informazioni sulle strutture dati utilizzate (i tipi) e sulle funzionalità delle classi inserite (metodi), informazioni utili al CLR per effettuare la compilazione JIT e gestire il programma in fase di esecuzione.
Il CLR, in base alle informazioni contenute nei Metadata presenti negli assembly ed in base alle impostazioni di sicurezza del sistema in cui è in esecuzione, si occupa del controllo dei tipi e dei permessi di esecuzione.

Altri compiti importanti del CLR sono:

- Controllare gli spazi di memoria da liberare, ovvero quelli che non sono più necessari all'esecuzione del programma (*Garbage Collector*);
- Gestione delle eccezioni;
- Verificare l'aderenza del codice con gli standard definiti dal CTS [7] (*Common Type System*) e CLS (*Common Language Specification*).

In conclusione il CRL è un modulo runtime univoco per tutti i linguaggi .NET che si occupa dell'esecuzione di tutto il codice. I compilatori .NET generano un linguaggio IL (Intermediate Language) che viene compilato JIT (Just in Time) prima di essere eseguito. E' importante sottolineare che IL non è interpretato ma convertito in codice macchina ed eseguito.

2.4.2.2 ASP.NET

ASP.NET è un framework per lo sviluppo di applicazioni dedicate al Web costruito sul Common Language Runtime.
ASP.NET è a sua volta suddiviso in tre componenti: L'ASP.NET Application (layer base), le ASP.NET Pages e ASP.NET Web Service (layer superiore).

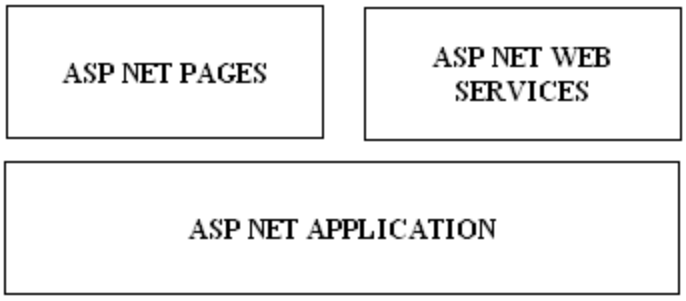


Figura 10: Divisione di ASP.NET nei suoi componenti

[7] Il CTS (*Common Type System*) si occupa della gestione unificata di tutti i tipi presenti nei vari linguaggi del Framework .NET, definendo come devono essere dichiarati, usati e gestiti dal runtime. E' possibile realizzare applicazioni utilizzando un qualunque linguaggio di programmazione che sia conforme alle specifiche indicate dal CTS.

L'applicazione rappresentata dall'intera soluzione rispecchia tale struttura, infatti essa è suddivisa in ASP.NET Pages, che implementano l'interfaccia utente, ed in Web Service che rendono disponibili all'applicazione delle funzionalità.
Durante il processo di sviluppo possiamo quindi distinguere due fasi: Web Forms e Web Service.

Le *Web Forms* in ASP.NET sono una tecnologia usata lato server per generare automaticamente pagine web. Queste pagine utilizzano dei controlli appositamente ideati tenendo presenti i seguenti obiettivi:

- La possibilità di creare ed utilizzare controlli UI riutilizzabili che possono incapsulare funzionalità comuni riducendo quindi la quantità di codice che uno sviluppatore di pagina deve scrivere;
- La possibilità per gli sviluppatori di strutturare in modo corretto ed ordinato la logica della pagina;
- La possibilità di rendere indipendente il lavoro del designer da quello dello sviluppatore.

Nello sviluppo di pagine ASP.NET si possono utilizzare i controlli server ASP.NET per programmare pagine Web. I controlli server vengono dichiarati all'interno di un file ASPX utilizzando tag personalizzati o tag HTML intrinseci, contenenti un valore di attributo *runat="server"*. I tag HTML intrinseci vengono gestiti da uno dei controlli nello spazio dei nomi *System.Web.UI.HTMLControls*. A qualsiasi tag, che non sia mappato in modo esplicito ad uno dei controlli, viene assegnato il tipo di *System.Web.UI.HTMLControls.HTMLGenericControl*.

Esempio 3: alcuni controlli server ASP.NET

```
<form runat=server>
<asp:textbox runat=server>
<asp:dropdownlist runat=server>
<asp:button runat=server>
```

In fase di esecuzione questi controlli server generano automaticamente contenuto HTML per il lato client.

Un *Web Service* è un'interfaccia che descrive una collezione di operazioni che sono accessibili via rete tramite messaggi XML in formato standard
Solitamente rappresenta un componente dell'applicazione che provvede ad una specifica funzionalità e comunica con il resto dell'applicazione attraverso i normali protocolli Web come HTTP, XML, SOAP, e UDDI.
I Web Service possono essere usati internamente alla singola applicazione od esposti esternamente su internet per poter essere usati da un numero qualsiasi di applicazioni.
Inoltre grazie al fatto che sono accessibili attraverso un'interfaccia standard, consentono a sistemi diversi di lavorare come un singolo sistema.

Esempio 4: esempio di Web Service

```
<%@ WebService Language="C#" Class="HelloWorld" %>

using System;
using System.Web.Service;

public class HelloWorld : WebService
{

      [WebMethod]
      public String SayHelloWorld()
      {
            return "Hello World";
      }

}
```

2.4.3 PocketPC

Un PocketPC è un piccolo computer particolarmente adatto alla multimedialità ed al training. Privo di tastiera, lo si usa tenendolo in palmo di mano e può essere comodamente contenuto nella tasca di una giacca o nella propria borsa. Con un PocketPC si può:

- Scrivere con Pocket Word o con Notes;
- Utilizzare un foglio di calcolo con Pocket Excel;
- Ascoltare musica in MP3 o WMA;
- Vedere filmati in formato WMV sfruttando il Media Player 9series;
- Connettersi e navigare in Internet con Pocket Internet Explorer;

- Leggere ed inviare posta elettronica;
- Con gli ultimi modelli si possono inoltre sfruttare tutte le potenzialità della connessione Wi-Fi.

Da un punto di vista Hardware le caratteristiche generali di questo dispositivo mobile sono le seguenti:

1. Processore Intel® fino a 400 MHz;
2. Sistema Operativo Windows® Mobile 2003 for PocketPC;
3. Memoria: quasi tutti i modelli sono equipaggiati con 32MB di memoria flash per il Sistema Operativo e per le applicazioni precaricate e 64 MB di SDRAM. Tutti i modelli hanno slot per espandere la memoria;
4. Display: TFT a matrice attiva da 3,5" fino 3,8"; 16-bit (65.000 colori) con risoluzione di 240 x 320;
5. Contrasto 10:1;
6. Audio: 16 bit stereo con altoparlanti e microfono integrato;
7. Tasti Speciali: fino a 6 tasti, rotella di scorrimento, interruttore a pressione, interruttore di bloccaggio, Pad direzionale;
8. Software: il software di base dei palmari è molto ampio e copre gran parte delle funzionalità di un comune PC. Ad esempio, per i PocketPC sono forniti: Microsoft® Pocket Outlook, Pocket Word, Pocket Excel, MSN® Messenger, Microsoft® Reader 2.0, Windows® Media Player 9.0 per Pocket PC, Voice Recorder, Calcolatrice, Pocket Explorer, ecc..;
9. Alimentazione: batteria ricaricabile Li-ion 8-10 ore di durata;
10. Dimensioni: da un minimo 115 x 70 x 12 fino a 130 x 80 x 13 mm;
11. Peso: 140-200 g;
12. Sistema di puntamento: *Touch Screen* per penna o tatto;
13. Connessione: inserendo una scheda Bluetooth SD e un cellulare si trasforma in un dispositivo *wireless*.

La caratteristica di questo dispositivo che va ad impattare principalmente nello sviluppo di una applicazione web è, senza alcun dubbio, la dimensione dello schermo a disposizione: l'area di lavoro effettivamente utilizzabile risulta infatti di 229x270 Pixels, a fronte di un display di 240x320 (decisamente ridotta rispetto ai comuni PC desktop).

Pocket Internet Explorer, per semplificare la navigazione e migliorare le prestazioni e l'utilizzo di memoria, permette l'apertura di un'unica finestra (ovvero non è possibile la visione di più pagine web contemporaneamente); questa caratteristica risulterà importante nella realizzazione del progetto PocketLezi in quanto andrà ad influenzare la fase di visualizzazione dei contenuti di un corso.
Per quanto riguardo il supporto ai *font*, tutti i dispositivi PocketPC includono almeno due stili: *Tahoma* e *Courier*. Tutti gli altri font vengono convertiti al più simile dei due prima menzionati, come definito dalla descrizione del loro font.
Un altro nodo cruciale riguarda la visualizzazione di immagini. Se le immagini incluse in una pagina web dovessero avere dimensioni maggiori rispetto all'area di lavoro, allora il browser si comporterà seguendo le seguenti regole:

- Se l'opzione *Fit-to-screen* è disabilitata, l'immagine verrà visualizzata come espresso dal tag <*img*>, rimanendo fedele agli attributi *height* e *width* forniti;
- Se l'opzione *Fit-to-screen* è attiva, la maniera in cui verrà visualizzata l'immagine dipenderà dalla sua dimensione, dalla *width* dell'elemento che contiene l'immagine e dalle attuali impostazione di dimensione carattere:
    - Se la dimensione dell'immagine è minore od uguale a quella del suo contenitore allora verrà visualizzata come espresso dal suo corrispondente tag <img>;
    - Se l'immagine è più grande del suo contenitore allora verrà effettuato automaticamente un ridimensionamento in modo da riempire esattamente tale contenitore; la lunghezza dell'immagine originale determinerà l'importo di tale ridimensionamento;
    - Le impostazioni di dimensione carattere impattano sulla percentuale di riduzione applicabile alle dimensioni originali di una immagine; le impostazioni di carattere "molto piccolo" rendono disponibili ridimensionamenti maggiori.

Pocket Internet Explorer implementa standard chiave per Internet: *Secure Sockets Layer* (SSL) per migliorare la sicurezza delle transazioni, supporto JScript (in versione proprietaria) per lo scripting all'interno delle pagine Web, cookies per la memorizzazione di dati locali, supporto ai fogli di stile CSS, supporto XHTML e HTML 4.0, supporto ai controlli ActiveX®.

Per lo sviluppo della nostra applicazione è stato utilizzato il Microsoft® Windows® Pocket PC 2003 SDK (*Software Development Kit*). Tale kit include un ambiente di emulazione che fornisce un computer virtuale con PocketPC 2003 compilato per i processore Intel® x86. Per maggiori informazioni su questo kit di sviluppo e sulle caratteristiche dei PocketPC si veda [MOBILE].

Per quanto riguarda i corsi fruibili da dispositivo mobile, proponiamo ora delle linee guida che dovrebbero essere considerate per poter rendere il più confortevole possibile la fruizione del corso stesso:

1. I titoli delle lezioni devono essere di lunghezza limitata (al massimo 20 caratteri tenendo conto degli spazi bianchi). Bisogna anche tenere presente che la lunghezza di tale titolo deve essere ridotta all'aumentare dell'annidamento della lezione.
2. Nella redazione dei contenuti tenere sempre presente che l'area effettivamente utilizzabile per la visualizzazione dei contenuti è all'incirca di 229x270 pixels. La cosa più importante è che la pagina non sia da "scorrere" in orizzontale, piuttosto esagerare nella lunghezza ma non sulla larghezza. Comunque non esagerare mai anche nello *scrolling* in verticale (diciamo al massimo due o tre schermate di contenuti).
3. Per le immagini si può tenere presente che il browser fornito insieme al Pocket adatta automaticamente le stesse (come già detto nelle pagine precedenti). Si consiglia comunque, per evitare qualsiasi problema, di non esagerare con le dimensioni delle stesse. Soprattutto si consiglia di inserire separatamente le immagini nel testo: con questo si intende che testo ed immagini non devono mai essere affiancate orizzontalmente ma seguire l'uno le altre e viceversa.
4. Per qualsiasi pagina di contenuti non utilizzare frame verticali; se proprio necessari solamente uno orizzontale ma niente di più (in quanto ridurre ulteriormente la superficie di visualizzazione sembra portare più svantaggi che altro).
5. Nel caso di corsi multimediali (quali ad esempio il corso di W2000) si pensano necessarie le seguenti modifiche:

I. Nella pagina della lezione visualizzare solamente il frame del video più i link verso altri siti oppure verso il materiale del corso; bisognerà inoltre tenere presente che nella versione Pocket di IE è sì possibile effettuare l'embed del Media Player all'interno di una pagina HTML, però il filmato deve essere compresso e reso disponibile in formato WMV con encoding per PocketPC.
II. Per ciò che concerne le *slides* si può pensare di fornire solo il link per scaricare il materiale; la visione all'interno della pagina, anche se possibile, sembra improponibile a causa delle dimensioni delle *slides* stesse.

3 Requisiti

Come già espresso nell'introduzione, l'applicazione PocketLezi deriva dall'applicazione Lezi.NET [1], un ambiente di produzione e fruizione di contenuti didattici pubblicabili in rete.
L'obiettivo primario del lavoro da noi svolto è rendere accessibile la parte di fruizione dei contenuti, per chi fosse pratico dell'architettura di Lezi.NET intendiamo il LeziViewer, a dispositivi mobili, in particolare i PDA con supporto Microsoft® PocketPC 2003. Le restrizioni inerenti il Sistema Operativo utilizzabile sul supporto mobile verranno chiarite più tardi nella sezione di "Analisi di Fattibilità", qui basti sapere che la scelta è stata dettata dal desiderio di espandere l'applicazione nell'ambito del multimodale (interazione vocale per il supporto agli utenti ipovedenti o non vedenti). Tale estensione richiederà il soddisfacimento di una serie di requisiti (di cui si parlerà più approfonditamente nel prossimo capitolo 3.1) tra i quali ricoprono particolare importanza:

1. *Contrazione/Amplificazione Spazio-Dimensionale, Sonora e Temporale*: ingrandimento, messa a fuoco, filtraggio, …;
2. *Trasduzione dei Contenuti*: Text to Speech e Speech to Text;
3. *Semplificazione dei Contenuti e della Navigazione*: insieme di servizi necessari per quegli utenti che necessitano di particolari aiuti durante la fase di apprendimento;
4. *Gerarchizzazione dei Contenuti per l'Interfacciamento Vocale*: per rendere più efficace l'interfacciamento vocale dell'applicazione.

Nell'ambiente Lezi.NET il contenuto fruibile non è altro che il corso, suddiviso in lezioni a loro volta suddivise in parti. Ogni parte è composta da una collezione di filmati o tracce audio, da una collezione di diapositive e da un insieme di documenti scaricabili ed hyperlink relativi al contesto della parte. L'applicazione Lezi.NET 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

[1] L'applicazione Lezi.NET è stata sviluppata in ambito di tesi dall'ing. Stefano Bruna

gestione dell'utenza, dei gruppi e dei permessi. Il nostro compito sarà quello di verificare la fattibilità di tutte le operazioni attualmente fruibili, sempre e solo nell'ambito di fruizione, e di riprogettarle in base alle caratteristiche hardware e software del nuovo dispositivo.
Riportiamo per comodità i requisiti generali del progetto:

- **Ambiente di fruizione per Mobile Device**: si vuole che un qualsiasi utente, fornito ovviamente di PDA, possa connettersi via web all'applicazione per fruire dei contenuti di un corso in maniera semplice e, dal punto di vista grafico, il più fedele possibile a quella dell'applicazione Lezi.NET;

- **Apertura all'interazione vocale**: prima della fase di progettazione è stato effettuato un lavoro di ricerca sulla tecnologie attuali per estendere l'applicazione (sia in versione PC che in versione PDA) nel campo dell'interazione vocale.

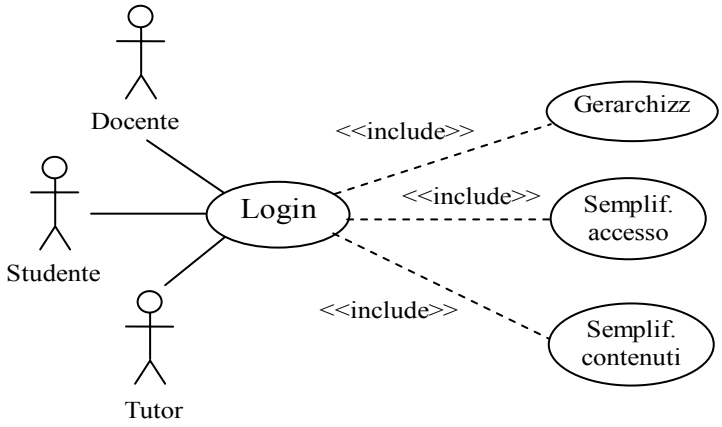
- **Adattabilità dei contenuti**: a seconda del device usato per connettersi all'applicazione, il sistema deve essere in grado di indirizzare e rendere disponibili all'utente solo quei corsi effettivamente fruibili dal device stesso;

- **Sincronizzazione dei contenuti**: il sistema deve tenere traccia del percorso di apprendimento di ciascun utente indipendentemente dal canale utilizzato per accedere all'applicazione, in modo da permettere il passaggio da un device all'altro senza dover ripetere le stesse lezioni più volte.

3.1 Funzionalità Generali

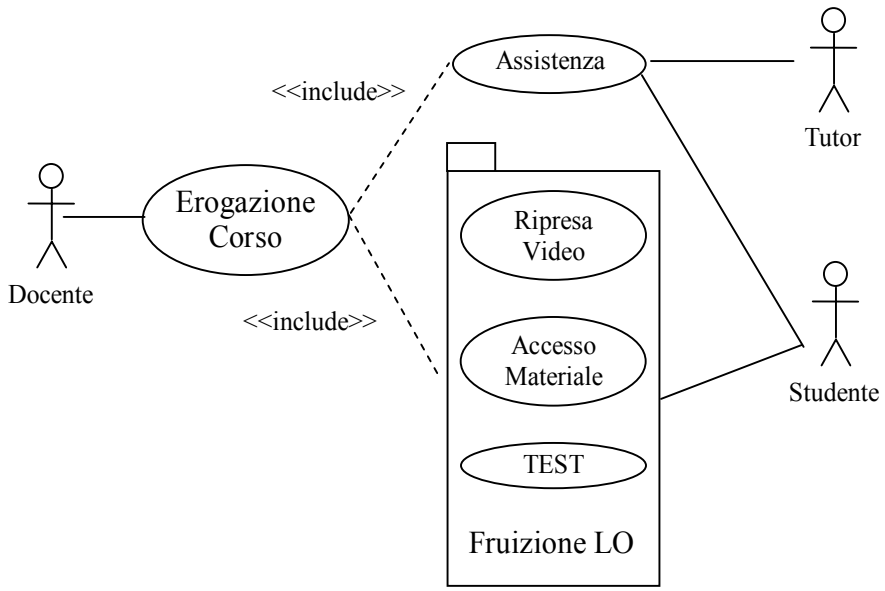
Riportiamo qui di seguito le funzionalità dell'intera estensione dell'ambiente Lezi.NET sia in ambito multicanale che multimodale. Vogliamo ricordare che tali funzionalità non esprimono i requisiti del progetto PocketLezi (cosa che avverrà nel capitolo 3.4) ma di un lavoro che coinvolge più gruppi.

1. ***Login***:

Nome	Login
Diagramma	
Descrizione	Qualsiasi operazione che avverrà in seguito presuppone una prima fase di Login in modo che il Sistema sia in grado di effettuare eventuali servizi di Semplificazione di Contenuti o di Navigazione per avere Accesso alle informazioni
Attori Principali	Studente, Tutor
Attori Secondari	Docente se vuole intervenire nell'erogazione del corso con ruolo di Assistenza
Pre-Condizioni	Utente registrato
Post-Condizioni	L'utente accede al sistema opportunamente adeguato al device utilizzato e al suo profilo
Associazioni	Include eventualmente "Semplificazione" (dei contenuti e dell'accesso alle informazioni) e "Gerarchizzazione dei Contenuti"
Requisiti Non	Si deve poter accedere al sistema con device diversi (PC, PDA

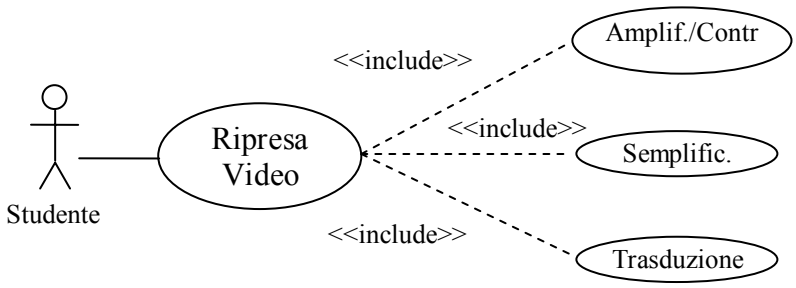
Funzionali	con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.
Input	Credenziali dell'Utente
Output	Accesso al Sistema con eventuali semplificazioni
Note	Tale semplificazione e gerarchizzazione avviene sia in base al profilo utente che si connette al sistema sia in base al device da esso utilizzato.

2. ***Erogazione Corso***:

Nome	Erogazione Corso
Diagramma	
Descrizione	Riguarda l'erogazione del corso da parte del Docente. Tale erogazione si struttura in una ripresa Video della lezione del docente, di un insieme di materiali didattici a cui lo studente può accedere, di una fase di assistenza a cui partecipano i vari Tutor e di un insieme di TEST atti a valutare la preparazione dello studente.
Attori Principali	Studente e Tutor
Attori Secondari	Docente con ruolo di Tutor (in quanto scopo di questa tesina è

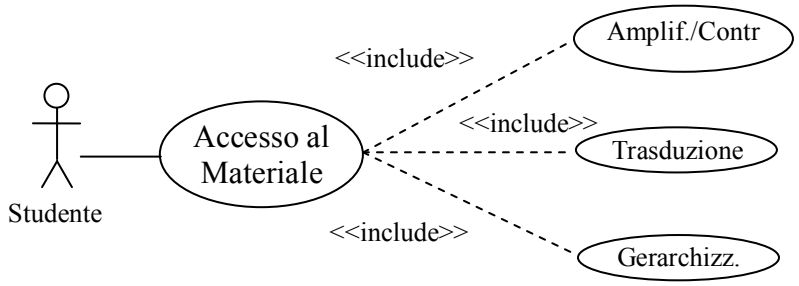
	l'ambiente di erogazione e non di creazione)
Pre-Condizioni	Studente registrato al Corso esistente
Post-Condizioni	Studente acquisisce conoscenza e una valutazione finale.
Associazioni	Include "Fruizione LO" ed "Assistenza".
Requisiti Non Funzionali	Si deve poter accedere al servizio con device diversi (PC, PDA con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.
Input	
Output	Valutazione dello Studente
Note	A seconda del profilo utente i vari sottoservizi si struttureranno a loro volta in altri servizi che verranno di seguito descritti. L'Assistenza è un servizio che deve essere sempre disponibile in ogni fase di fruizione del LO.

3. ***Ripresa Video***:

Nome	Ripresa Video
Diagramma	
Descrizione	Lo studente visualizza sul proprio device la ripresa video della lezione del docente. Tale visione può avvenire sia in maniera Asincrona (a lezione già tenuta e video registrata) che Sincrona (durante lo svolgimento della lezione vera e propria).
Attori Principali	Studente
Attori Secondari	Docente che tiene la lezione

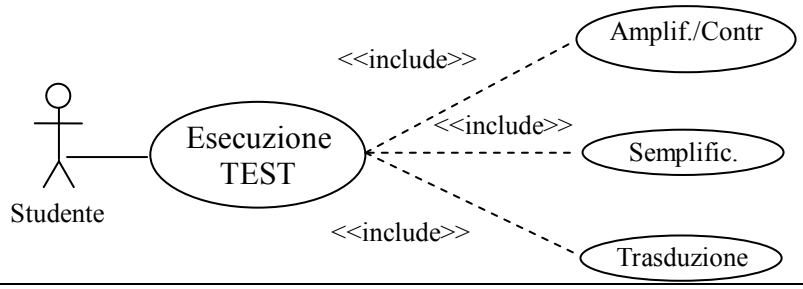
Pre-Condizioni	Studente in modalità erogazione corso a cui è registrato
Post-Condizioni	Lo studente visualizza la ripresa video della lezione del docente
Associazioni	Include "Amplificazione/Contrazione" (spaziale-dimensionale, sonora, temporale) e "Trasduzione dei Contenuti" (Speech to Text). E' incluso in "Erogazione Corso"
Requisiti Non Funzionali	Si deve poter accedere al servizio con device diversi (PC, PDA con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.
Input	L'utente richiede la visione di un corso
Output	Video del docente
Note	Con Amplificazione/Contrazione si intende: • Spaziale - Dimensionale: Ingrandimento, Messa a Fuoco, etc.. • Sonora: Amplificazione, Filtraggio, Trasposizione in Frequenza, etc.. • Temporale: Contrazione, Dilatazione, etc.. A tale proposito bisogna tenere presente degli effetti di disturbo che la ripresa video (e quindi audio) avrà sull'interfacciamento vocale al sistema.

4. ***Accesso al Materiale***:

Nome	Accesso al Materiale
Diagramma	
Descrizione	Lo studente accede al materiale del corso (LO) messo a

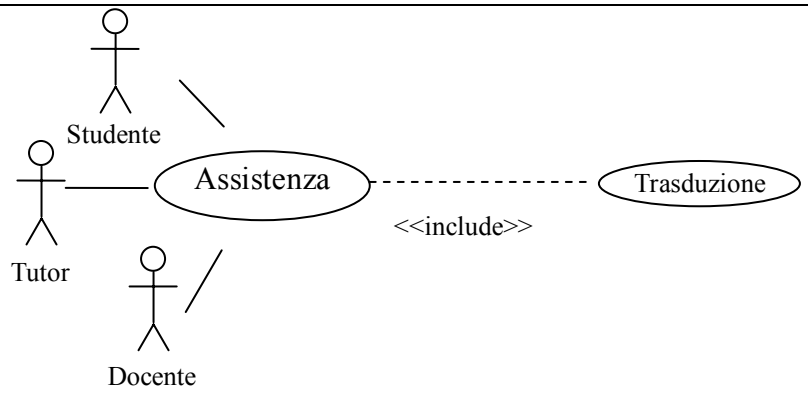
	disposizione dal docente (diapositive PowerPoint, contenuto di una lavagna luminosa o tradizionale o episcopio.
Attori Principali	Studente
Attori Secondari	Docente come fornitore del materiale
Pre-Condizioni	Studente registrato al corso il cui materiale è fornito dal docente e quindi pubblicato
Post-Condizioni	Lo studente ottiene il materiale desiderato
Associazioni	Include "Amplificazione/Contrazione" (spaziale-dimensionale), "Trasduzione dei Contenuti" (Text to Speech) e "Gerarchizzazione dei Contenuti per Interfacciamento Vocale". E' incluso in "Erogazione Corso".
Requisiti Non Funzionali	Si deve poter accedere al servizio con device diversi (PC, PDA con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.
Input	L'utente seleziona il materiale (LO) desiderato
Output	L'utente ottiene il materiale (LO)
Note	

5. ***Esecuzione Test***:

Nome	Esecuzione TEST
Diagramma	
Descrizione	Una volta assistito alla lezione o durante la stessa, lo studente eseguirà dei TEST per poterne valutare l'apprendimento.
Attori Principali	Studente

Attori Secondari	Docente come fornitore del TEST
Pre-Condizioni	Studente iscritto ad un corso
Post-Condizioni	Ottenere una valutazione sullo stato di apprendimento dello studente
Associazioni	Include "Amplificazione/Contrazione" (spaziale-dimensionale), "Trasduzione dei Contenuti" (Text to Speech e Speech to Text). E' incluso in "Erogazione Corso".
Requisiti Non Funzionali	Si deve poter accedere al servizio con device diversi (PC, PDA con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.
Input	Risposte al TEST fornite dallo studente
Output	Valutazione del TEST
Note	

6. ***Assistenza***:

Nome	Assistenza
Diagramma	
Descrizione	Riguarda l'attività di assistenza svolta dai Tutor per supportare l'apprendimento degli studenti.
Attori Principali	Studente, Tutor
Attori Secondari	Docente con ruolo di Tutor.
Pre-Condizioni	Studente registrato al corso
Post-Condizioni	Lo studente ottiene spiegazioni aggiuntive e chiarimenti inerenti la

	lezione
Associazioni	Include "Trasduzione dei Contenuti" (Text to Speech e Speech to Text). E' incluso in "Erogazione Corso" ed accessibile in qualsiasi momento di tale erogazione.
Requisiti Non Funzionali	Si deve poter accedere al servizio con device diversi (PC, PDA con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.
Input	Messaggio dello studente e/o del Tutor
Output	Messaggio del Tutor e/o dello studente
Note	Tale fase di Assistenza comprende anche la possibilità di più studenti, fruenti lo stesso corso, di discutere tra di loro eventuali problemi ed approfondimenti.

7. ***Amplificazione/Contrazione***:

Nome	Amplificazione/Contrazione
Diagramma	Vedi i diagrammi precedenti.
Descrizione	Modalità di interazione messe a disposizione dei vari utenti per facilitare l'utilizzo dell'applicazione. Con Amplificazione/Contrazione si intende: • Spaziale-Dimensionale: Ingrandimento, Messa a Fuoco, etc.. • Sonora: Amplificazione, Filtraggio, Trasposizione in Frequenza, etc.. • Temporale: Contrazione, Dilatazione, etc..
Attori Principali	Studente, Tutor e Docente
Attori Secondari	
Pre-Condizioni	Utente registrato al corso e con profilo utente che permetta tali servizi

Post-Condizioni	
Associazioni	E' incluso in "Erogazione Corso" e in tutti i suoi sottoservizi.
Requisiti Non Funzionali	Si deve poter accedere al servizio con device diversi (PC, PDA con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.
Input	Selezione della modalità desiderata
Output	Adeguamento del servizio in base alla selezione effettuata
Note	Bisogna tenere presente delle interferenze che tali modifiche apporteranno alla fruizione del corso (in particolare il problema della sincronizzazione)

8. ***Trasduzione dei Contenuti***:

Nome	Trasduzione dei Contenuti
Diagramma	Vedi i diagrammi precedenti.
Descrizione	Modalità di interazione messe a disposizione dei vari utenti per facilitare l'utilizzo dell'applicazione. Con Trasduzione dei Contenuti si intende: • Text to Speech • Speech to Text
Attori Principali	Studente, Tutor e Docente
Attori Secondari	
Pre-Condizioni	Utente registrato al corso e con profilo utente che permetta tali servizi
Post-Condizioni	
Associazioni	E' incluso in "Erogazione Corso" e in tutti i suoi sottoservizi.
Requisiti Non Funzionali	Si deve poter accedere al servizio con device diversi (PC, PDA con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.

Input	Qualsiasi operazione che necessiti di tale trasduzione
Output	Trasduzione desiderata
Note	Bisogna tenere presente delle interferenze che tali modifiche apporteranno alla fruizione del corso (in particolare il canale audio potrebbe già essere occupato dalla ripresa video della lezione).

9. ***Semplificazione dei Contenuti/Navigazione***:

Nome	Semplificazione dei Contenuti/Navigazione
Diagramma	Vedi i diagrammi precedenti.
Descrizione	Insieme di servizi necessari per quegli studenti che necessitano di particolari aiuti durante la fase di apprendimento e per quegli studenti che si interfacciano col sistema per mezzo di device con precise necessità. Con Semplificazione si intende: • Semplificazione dell'interazione (interfacciamento) con il sistema • Semplificazione dei contenuti • Semplificazione della navigazione e dell'accesso alle informazioni
Attori Principali	Studente
Attori Secondari	
Pre-Condizioni	Utente iscritto al corso e con profilo utente che necessiti di tali semplificazioni; utente iscritto ad un corso ed in fase di fruizione dello stesso
Post-Condizioni	Facilitazione dei contenuti e degli schemi navigazionali
Associazioni	E' incluso in "Login" e in "Erogazione Corso".
Requisiti Non Funzionali	Si deve poter accedere al servizio con device diversi (PC, PDA con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.

Input	Dati di un utente fornito di profilo che necessiti tali semplificazioni oppure richiesta di semplificazione da parte di un qualsiasi utente durante la fase di fruizione del corso
Output	Semplificazione dei contenuti e degli schemi navigazionali.
Note	

10. ***Gerarchizzazione dei Contenuti per l'Interfacciamento Vocale***:

Nome	Gerarchizzazione dei Contenuti per l'Interfacciamento Vocale
Diagramma	Vedi i diagrammi precedenti.
Descrizione	Tale gerarchizzazione serve per rendere più efficace l'interfacciamento vocale all'applicazione
Attori Principali	Docente, Studente, Tutor
Attori Secondari	
Pre-Condizioni	Utente iscritto al corso e con un profilo utente che necessiti interfacciamento vocale oppure utente che si collega al sistema con un particolare device che necessita sempre di interfacciamento vocale
Post-Condizioni	
Associazioni	E' incluso in "Login" e in "Accesso al Materiale"
Requisiti Non Funzionali	Si deve poter accedere al servizio con device diversi (PC, PDA con abilitazione telefonica, Tablet PC) sia in modalità sincrona che asincrona. L'accesso oltre che multicanale deve essere anche multimodale.
Input	
Output	
Note	

3.2 Analisi di Fattibilità

Prima di procedere all'implementazione dell'applicazione PocketLezi, abbiamo rivolto la nostra attenzione ad uno studio approfondito inerente la realizzazione di interfacce vocali e multimodali (in cui si accoppiano parte grafica e parte vocale). L'obiettivo è quello di estendere in futuro la piattaforma di Lezi.NET e PocketLezi per renderle accessibili ad un'utenza ipovedente e non vedente.

Una prima parte di ricerca è dedicata ai linguaggi attualmente affermatisi in questo ambito. Tale ricerca ha portato essenzialmente a due risultati: *VoiceXML* (o meglio *XHTML+Voice*) e *SALT*. Nei prossimi paragrafi parleremo più approfonditamente di questi linguaggi, comunque, ad una prima analisi, sebbene in competizione, tali tecnologie sembrano ottenere i risultati migliori in ambiti separati: *VoiceXML* rende al meglio nelle applicazioni puramente vocali, *SALT* ottiene i risultati migliori nella realizzazione di applicazioni multimodali.

3.2.1 VoiceXML (XHTML+Voice)

VoiceXML (*Voice eXtensible Markup Language*) è un linguaggio di *markup* per la creazione di interfacce utenti vocali. Per effettuare l'inserimento dati usa il riconoscimento vocale e/o toni DTMF (Dual Tone Multi-Frequency), mentre per l'output usa audio pre-registrato e sintesi Text-to-Speech (TTS).

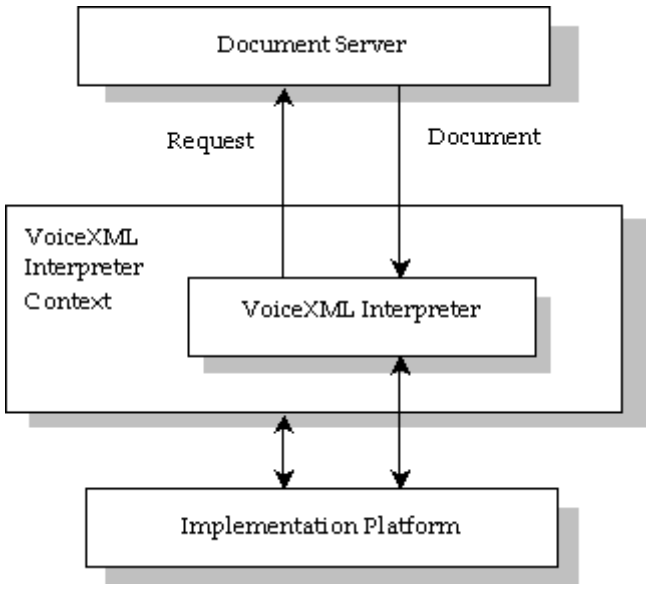


Figura 11: Modello Architetturale VoiceXML

Un *Document Server* (per esempio un server Web) processa le richieste da una applicazione client, il *VoiceXML Interpreter*, attraverso il *VoiceXML Interpreter Context*. Il server produce in risposta documenti VoiceXML che vengono interpretati dal VoiceXML Interpreter. Il VoiceXML Interpreter Context può monitorare gli input provenienti da un utente in parallelo al VoiceXML Interpreter.

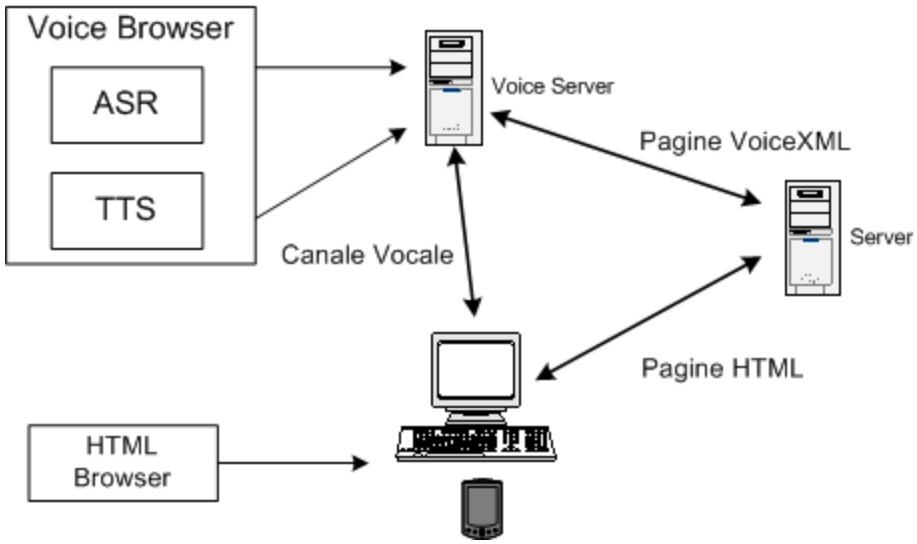


Figura 12: Architettura Implementativa VoiceXML

Il *Voice Server* è il componente che consente di realizzare la comunicazione vocale con l'utente assolvendo alla funzionalità di browser vocale, interpretando le pagine VoiceXML create dal server per tutti i dispositivi telefonici (che non hanno un proprio browser vocale). Al suo interno sono implementati il TTS (Text-to-Speech) e l'ASR (Automatic Speech Recognition).
La soluzione proposta consente all'utente di interagire attraverso un'interfaccia grafica ed una vocale; è possibile accedere a tali interfacce utilizzando due browser che permettano l'interpretazione delle pagine generate in HTML e VoiceXML:

- *HTML Browser* consente al device di visualizzare le pagine HTML implementando quindi l'interfaccia grafica del servizio. È un tradizionale browser HTML, come ad esempio Internet Explorer, il cui unico requisito è il supporto alle funzioni JavaScript, necessarie per poter effettuare la sincronizzazione degli input e degli output. Questo browser è l'unico componente che deve essere installato sui terminali per il corretto funzionamento della piattaforma.

- *Voice Browser*, generalmente installato su di un Voice Server, consente l'interpretazione delle pagine VoiceXML create per l'interazione vocale. La piattaforma produce pagine aderenti alle specifiche VoiceXML 2.0.

Problema di sincronizzazione tra i due browser: a tal fine è necessario progettare un componente in grado di creare pagine HTML e VoiceXML che, periodicamente a scadenze temporali definite e configurabili, effettuino richieste di *polling* al server ed eventualmente aggiornando la pagina attiva sullo specifico canale. Per risolvere tale problema di sincronizzazione si potrebbe optare per l'utilizzo di un browser multimodale ovvero che incorpori automaticamente l'interazione vocale e quella grafica. Il più diffuso e compatibile con le specifiche XHTML + VoiceXML è il browser Opera il quale però è sfornito di pieno supporto al linguaggio JScript (linguaggio pesantemente utilizzato nella applicazione Lezi.NET già esistente e funzionante). Tale mancanza di supporto potrebbe essere risolta con l'installazione di qualche Plug-In ma il completo funzionamento della applicazione non può essere comunque garantito.

Per ulteriori approfondimenti sulla tecnologia e linguaggio VoiceXML si rimanda a [VoiceXML] e [W3C.VoiceXML].

3.2.2 SALT

SALT (*Speech Application Language Tags*) estende gli esistenti linguaggi di markup quali HTML, XHTML ed XML abilitando l'accesso multimodale dell'utente e consentendo di interagire con l'applicazione in diverse modalità indipendentemente e concorrenzialmente.
Consiste di un piccolo set di elementi XML ai quali sono associati attributi, proprietà, eventi e metodi (attraverso il Document Object Model); tali elementi sono utilizzabili per applicare un'interfaccia vocale alle pagine web. SALT può essere utilizzato con HTML, XHTML ed altri standard sia per interfacce interamente vocali (ad esempio telefoni), sia per applicazioni multimodali.

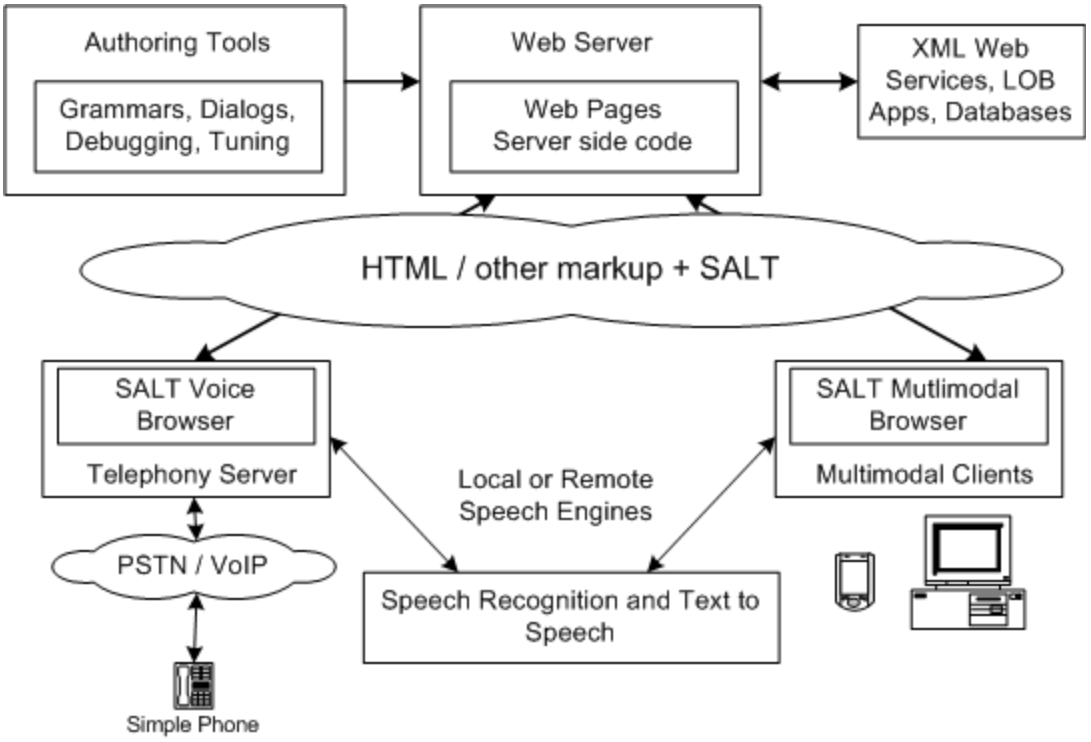


Figura 13: Architettura di Riferimento SALT

Anche in questo caso esistono dei browser SALT-compatibili (per ulteriori approfondimenti vedere [SALT]) oppure dei Plug-In per Internet Explorer scaricabili dal sito Microsoft (questa possibilità potrebbe risolvere i problemi di incompatibilità con l'architettura di Lezi.NET già esistente).
Sicuramente sappiamo che:

- I telefoni tradizionali senza processore dovranno effettuare una telefonata ad un server su cui è in esecuzione un voice-only SALT browser;
- I PC tradizionali o comunque i Tablet PC/Notebook potranno effettuare i riconoscimenti vocali e i processi di output vocali localmente;
- I device più piccoli, con limitate capacità di calcolo (quali ad esempio i PDA) avranno installato un browser SALT compatibile ma useranno server remoti per il riconoscimento vocale o per i servizi di sintesi.

L'aspetto che rende questa tecnologia preferibile rispetto al VoiceXML (almeno per quanto riguarda la multimodalità) è l'esistenza di un ambiente di sviluppo integrabile in

Visual Studio® .NET 2003. Visto che l'intera applicazione Lezi.NET è stata sviluppata in tale ambiente, si risolvono tutti i problemi di sincronizzazione dei browser e di incompatibilità di linguaggi.
Microsoft® Speech Server contiene una soluzione completa per sviluppare, testare e pubblicare applicazioni telefoniche (speech only) e multimodali (visuale + voce). Tale prodotto è composto essenzialmente da: *Microsoft® Speech Server* (*MSS*) e *Microsoft® Speech Application SDK* (*SASDK*).

Il MSS contiene tutti i componenti server per la pubblicazione di applicazioni telefoniche e multimodali. MSS funziona in ambiente Windows Server™ 2003 e permette il riconoscimento vocale e la sintesi vocale per telefoni fissi, telefoni cellulari e dispositivi PocketPC. Per le applicazioni telefoniche, MSS include dei servizi per connettersi ad un *PBX* privato (*Private Branch Exchange*) e linee telefoniche in configurazioni molto flessibili.

Microsoft® .NET Speech SDK (al momento giunto alla versione Beta 4) soddisfa la necessità degli sviluppatori di applicazioni vocali di API, controlli ed applicativi per l'estensione di Visual Studio® .NET nel dominio dell'interazione vocale.
L'*SASDK* (*Speech Application Software Development Kit*) include gli *speech add-in* lato client, Speech Add-in per Microsoft® Internet Explorer 6.0 e Speech Add-in per Microsoft Pocket Internet Explorer (questi ultimi sono realizzati per essere eseguiti in ambiente PocketPC 2003); tramite queste utility i PC Desktop, Tablet PC e Pocket PC saranno in grado di comprendere i tag vocali inclusi all'interno del codice HTML come definito dalle specifiche dello standard SALT. Per maggiori approfondimenti si rimanda a [SASDK].

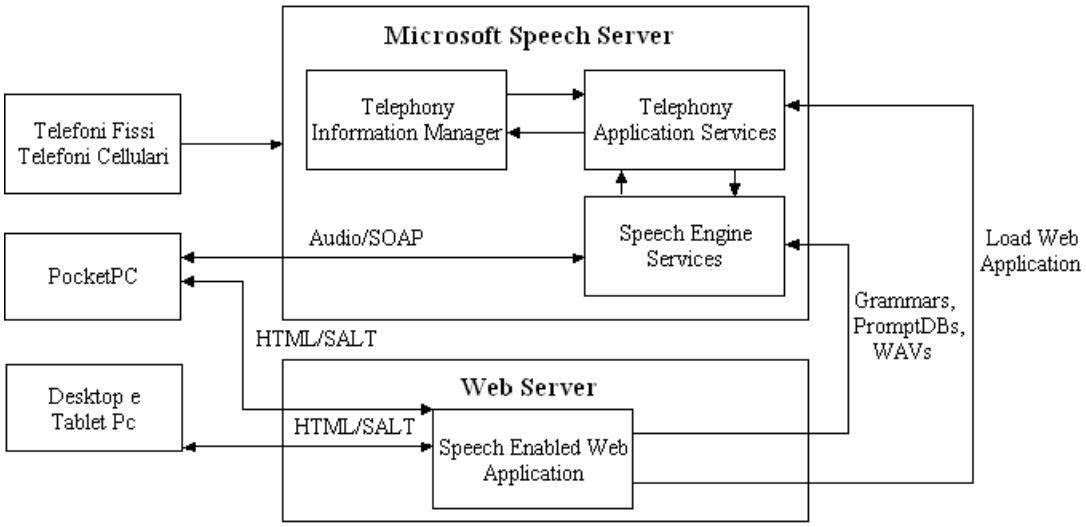


Figura 14: Componenti principali della soluzione Microsoft® Speech Server

Riassumiamo nella tabella seguente i componenti necessari per la realizzazione di applicazioni con tecnologia Microsoft® Speech Server:

Tabella 1: Componenti necessari per lo sviluppo di applicazioni telefoniche e multimodali

Applicazione (Device Supportati)	Componenti necessari
Telefoni (fissi, cellulari)	**Componenti Microsoft Speech** • Speech Engine Service (SES) • Telephony Application Service (TAS) • Telephony Interface Manager (TIM) **Altri componenti richiesti** • Adattatore telefonico supportato per la connessione di linee telefoniche o sistema PBX • Microsoft Windows Server 2003 con Internet Information Service • Distributori di carico (opzionale per carichi distribuiti a SES e Web Server multipli) • PBX (opzionale per la progettazione di componenti TAS distribuiti)
Multimodale (Pocket PC)	**Componenti Microsoft Speech** • Speech Engine Service (SES) • Speech Add-in per Microsoft Pocket Internet Explorer **Altri componenti richiesti** • Microsoft Windows Server 2003 con Internet Information Service • Distributori di carico (opzionale per carichi distribuiti a SES e Web Server multipli)
Multimodale (desktop PC, Tablet PC)	**Componenti Microsoft Speech** • Speech Add-in per Microsoft Internet Explorer

	Altri componenti richiesti • Microsoft Windows 2003 con Internet Information Service • Distributori di carico (opzionale per carichi distribuiti a SES e Web Server multipli)

3.3 Requisiti

Riportiamo ora il diagramma UML rappresentante i requisiti e le funzionalità che l'applicazione PocketLezi deve soddisfare. Di seguito proporremo una serie di tabelle che analizzeranno i singoli requisiti (quelli più significativi) con più precisione.

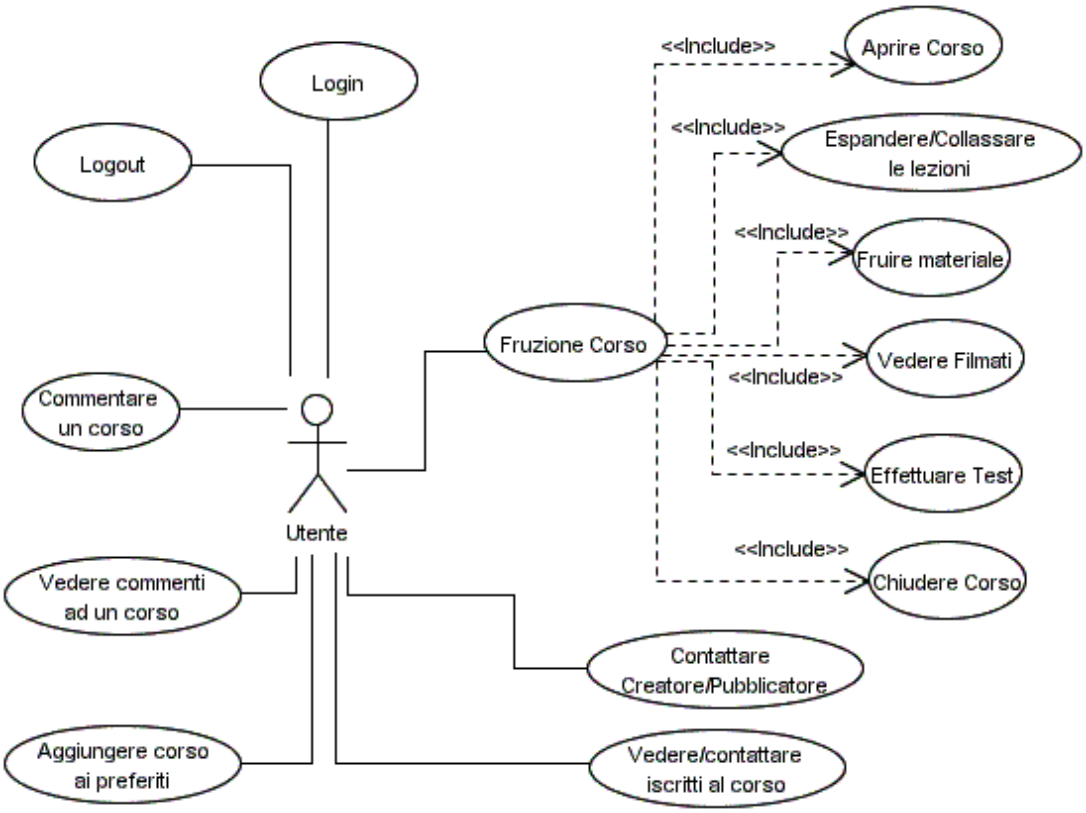


Figura 15: Use Case Diagram riportante i requisiti dell'applicazione PocketLezi

Tabella 2: Use Case "Aprire Corso"

Use Case	Aprire Corso
Pre-Condizione	Utente identificato con successo
Percorso Base	1. L'utente inserisce login e password nella pagine di autenticazione; 2. L'utente seleziona uno dei corsi dal menu visualizzato.
Post-Condizione	L'utente entra nella pagina con l'albero dei contenuti

Tabella 3: Use Case "Espandere/Collassare le Lezioni"

Use Case	Espandere/Collassare le Lezioni
Pre-Condizione	1. Utente identificato; 2. L'utente ha selezionato uno dei corsi disponibili; 3. Utente nell'ambiente di fruizione del corso.
Percorso Base	1. L'utente seleziona una delle icone apposite dall'albero dei contenuti (il segno "+" per espandere, il segno "-" per collassate).
Post-Condizione	L'albero dei contenuti si espande o meno a seconda della scelta effettuata

Tabella 4: Use Case "Fruire Materiale/ Vedere Filmati"

Use Case	Fruire Materiale e Vedere Filmati
Pre-Condizione	1. Utente identificato; 2. L'utente ha selezionato uno dei corsi disponibili; 3. Utente nell'ambiente di fruizione del corso.
Percorso Base	1. L'utente seleziona una delle lezioni disponibili dall'albero dei contenuti corrispondente al corso scelto; 2. Il sistema visualizza la pagina del contenuto; tale contenuto potrà essere una pagina HTML oppure un filmato a seconda del tipo di corso pubblicato;

	3. 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.
Post-Condizione	Il sistema tiene traccia dell'apprendimento dell'utente.

Tabella 5: Use Case "Effettuare Test"

Use Case	Effettuare Test
Pre-Condizione	1. Utente identificato; 2. L'utente ha selezionato uno dei corsi disponibili; 3. L'utente nell'ambiente di fruizione del corso.
Percorso Base	1. L'utente seleziona dall'albero dei contenuti la lezione corrispondente al Test di verifica; 2. Il sistema visualizza la pagina del contenuto; tale contenuto potrà essere una pagina HTML oppure un filmato a seconda del tipo di corso pubblicato; 3. 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.
Post-Condizione	Il sistema tiene traccia dell'apprendimento dell'utente.

Tabella 6: Use Case "Chiudere Corso"

Use Case	Chiudere Corso
Pre-Condizione	1. Utente identificato; 2. L'utente ha selezionato uno dei corsi disponibili.
Percorso Base	1. L'utente seleziona l'apposito link di chiusura corso.
Post-Condizione	L'utente viene riportato alla pagina di default.

Tabella 7: Use Case "Contattare Altri Utenti"

Use Case	Contattare altri utenti
Pre-Condizione	1. Utente identificato; 2. L'utente ha selezionato uno dei corsi disponibili.
Percorso Base	1. L'utente seleziona dal menu l'elenco degli iscritti al corso; 2. Nella tabella degli iscritti al corso l'utente seleziona il link per visualizzare i dettagli; 3. L'utente seleziona l'indirizzo di posta elettronica dell'utente desiderato.
Post-Condizione	Si apre il client predefinito per l'invio di posta elettronica.

Tabella 8: Use Case "Contattare Creatore e/o Pubblicatore del Corso"

Use Case	Contattare creatore e/o pubblicatore del corso
Pre-Condizione	1. Utente identificato; 2. L'utente ha selezionato uno dei corsi disponibili.
Percorso Base	1. L'utente seleziona dal menu l'operazione desiderata.
Post-Condizione	Si apre il client predefinito per l'invio di posta elettronica.

Tabella 9: Use Case "Commentare un Corso"

Use Case	Commentare un Corso
Pre-Condizione	1. Utente identificato; 2. L'utente ha selezionato uno dei corsi disponibili.
Percorso Base	1. L'utente seleziona dal menu l'apposita operazione; 2. L'utente compila la form apposita per l'operazione; 3. L'utente conferma l'inserimento del commento.
Post-Condizione	Il sistema manda un messaggio di conferma di inserimento.

4 Progettazione

4.1 Ruoli, Gruppi ed Utenti

L'applicazione Lezi.NET, per poter associare un ruolo ad un utente, realizza un modello *role-based* basato su gruppi ed utenti simile a quello che si può trovare in molti sistemi operativi multi-utente. Il sistema LMS 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 guest).
- **Users**: gruppo che contiene gli utenti registrati al sistema. 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.

Nell'ambito del progetto da noi svolto tale distinzione di gruppi e ruoli non interviene così pesantemente come nel progetto padre, in quanto l'ambito di fruizione risulta lo stesso per tutti i gruppi e per tutti gli utenti. Unica caratteristica leggermente diversa è l'interfaccia grafica che disabilita alcune operazioni a seconda dei diritti dell'utente di volta in volta connesso. Per questo nei diagrammi che seguiranno (in particolare gli *Use Case Diagram*) l'utente verrà indicato genericamente con l'identificatore Utente e non con il proprio ruolo che lo rende membro di uno dei quattro gruppi definiti.

4.2 Classi e Strutture Dati

Per la creazione delle classi che modellano le entità coinvolte si è rimasti totalmente fedeli allo schema presentato dall'applicazione Lezi.NET. Questa scelta, oltre che permetterci un grosso risparmio di tempo dal punto di vista implementativo, ci permette di evitare qualsiasi tipo di incompatibilità tra le due piattaforme, cosa che risulterà di particolare importanza nella parte di Sincronizzazione dei Contenuti (vedi Cap. 5.2).

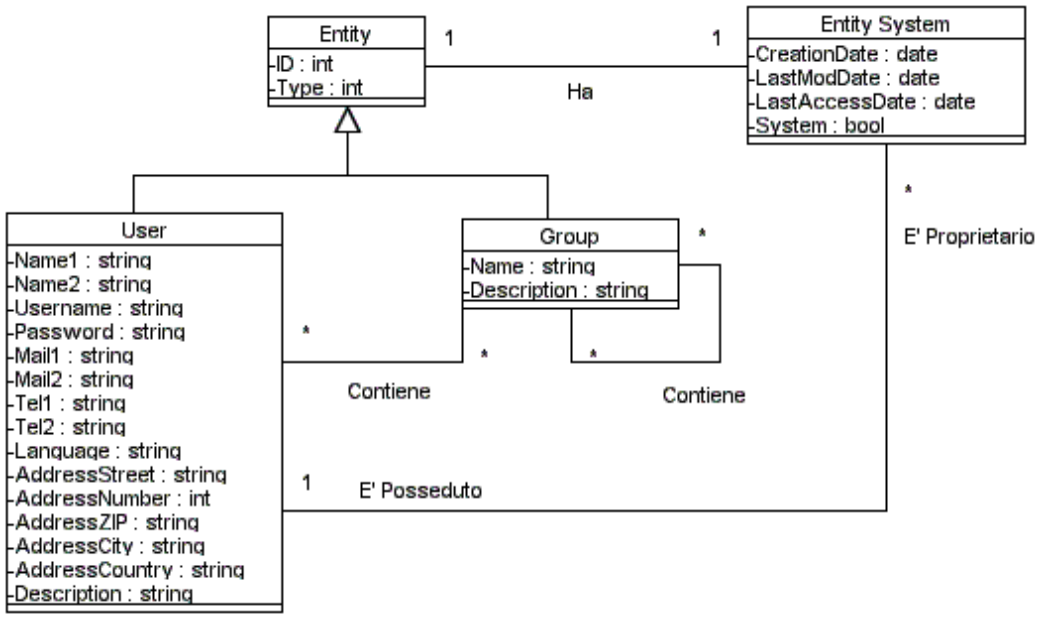


Figura 16: Diagramma delle classi inerenti utenti e gruppi.

La classe Entity è padre delle classi User e Group. In tal modo si ha un unico identificatore per le diverse istanze di User e Group, questo per 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.

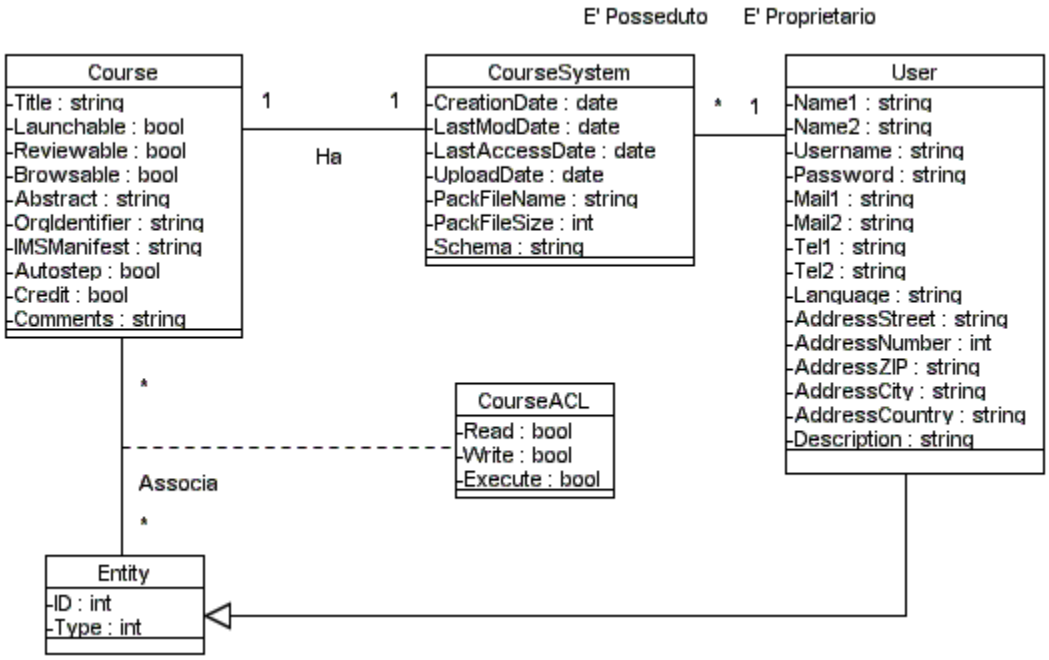


Figura 17: Diagramma delle classi inerenti la relazione tra utenti, gruppi e corsi.

Anche per la classe Course le informazioni relative al sistema sono state separate in una classe 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 utenti 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. Un'esigenza implementativa di questo tipo condiziona la fase di progettazione. Tenendo conto di questo aspetto il diagramma di Figura 17 può essere esteso nel diagramma di Figura 18 introducendo la classe Server e le sue relazioni con le altre entità .

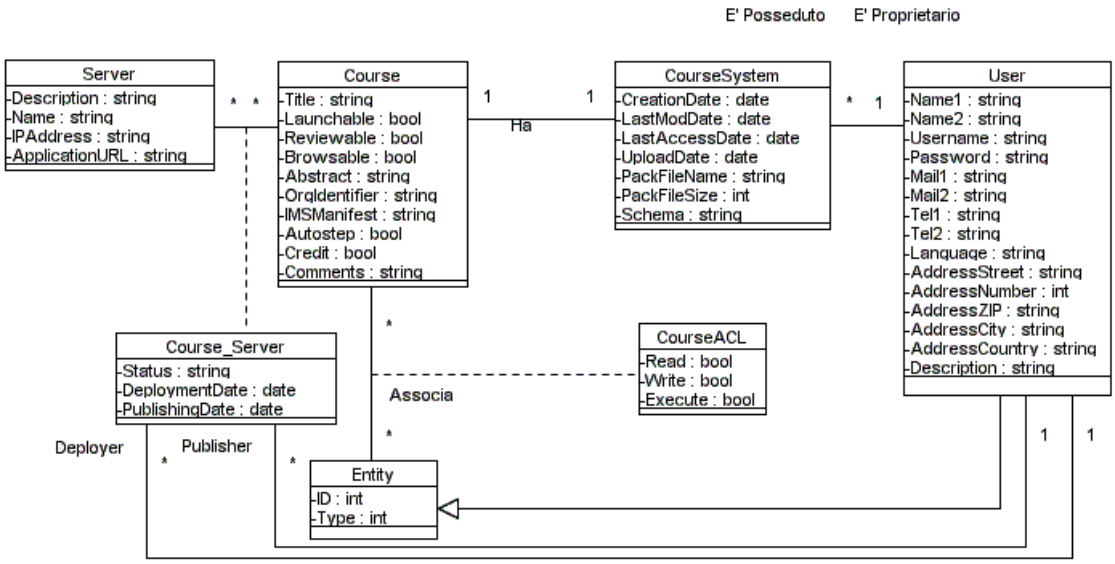


Figura 18: Diagramma delle classi tenendo conto della possibilità di avere più delivery Service.

Per supportare i corsi SCORM e tenere traccia delle interazioni dell'utente con gli SCO costituenti i corsi, 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.

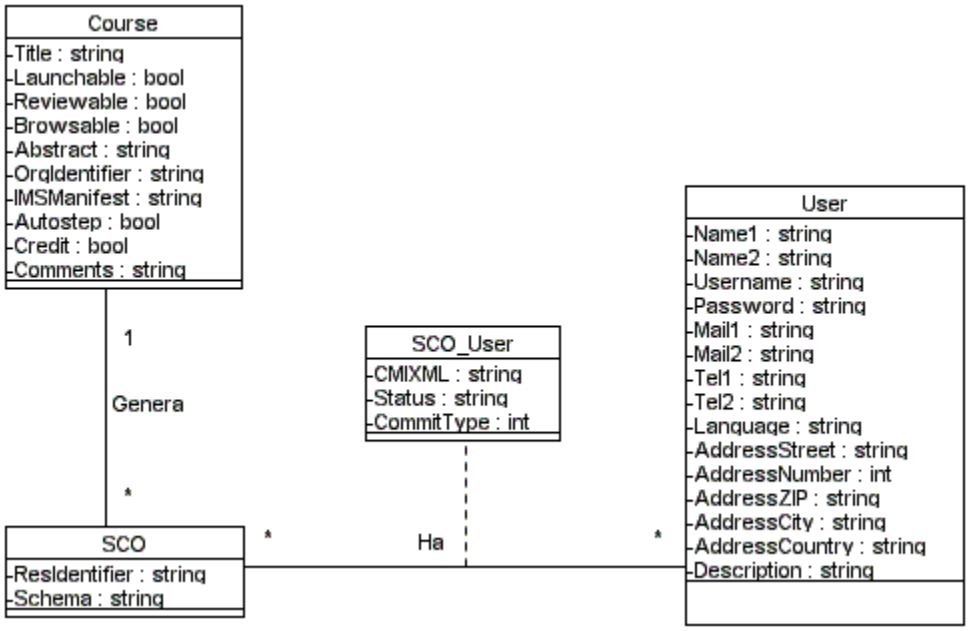


Figura 19: Diagramma delle classi con la relazione fra utente e SCO per tracciare l'apprendimento

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 Object Oriented è il modello CMI di cui si è parlato nella Sezione 2.3.2.4. Un modo naturale di rappresentare la struttura dati che implementa CMI è un documento XML.
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 delle 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 foglie 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 cardinalità dei rami della struttura;
- Aggiornamento dei campi che contengono contatori per i rami aventi cardinalità variabile;
- Controllo dei passaggi di stato di alcuni elementi con particolari vincoli sui valori;
- 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;
```

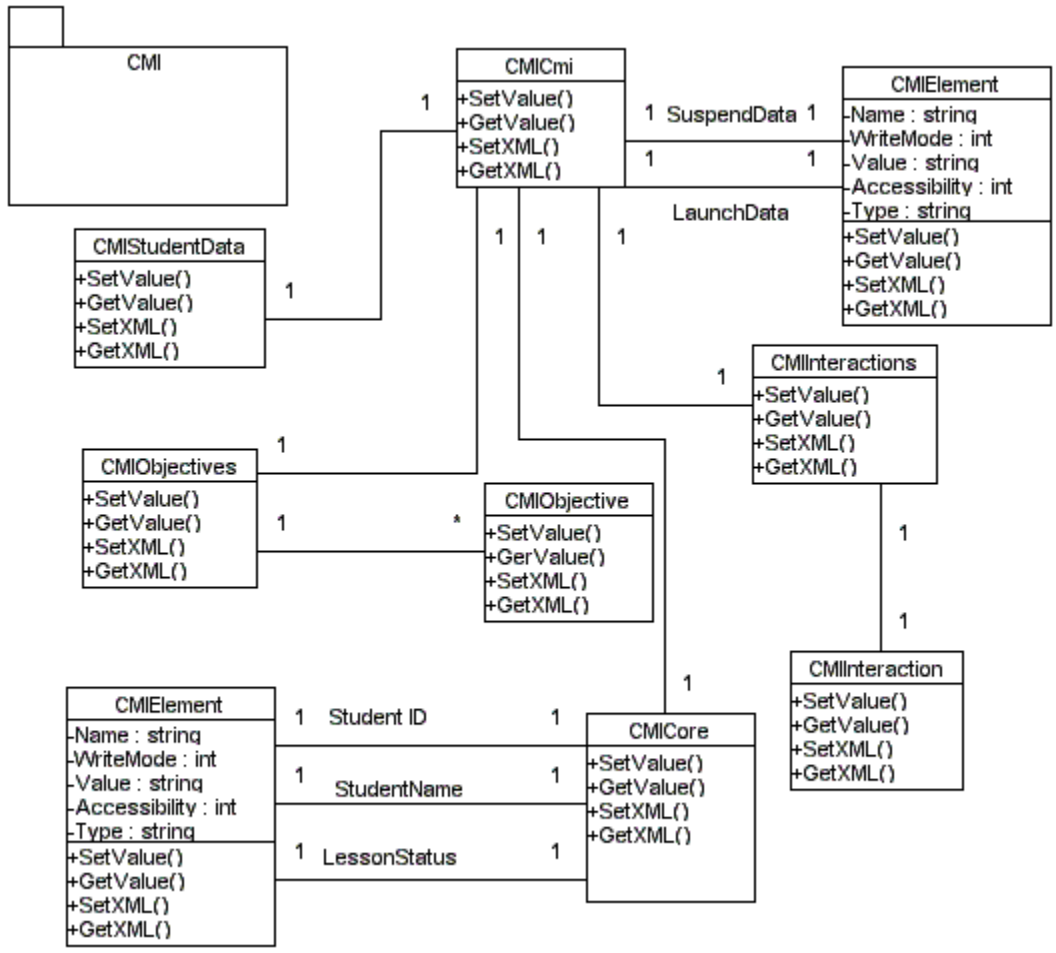


Figura 20: Diagramma parziale delle classi del Package CMI; tutti gli elementi foglia della struttura CMI sono rappresentati dalla classe CMIElement che viene inizializzato diversamente dal padre a seconda del tipo di dato che dovrà contenere.

Struttura IMSManifest: l'applicazione Lezi.NET fornisce un'API, funzionante in sola lettura, per poter avere un'interfaccia Object Oriented al documento imsmanifest.xml descritto dettagliatamente nella sezione 2.3.1.3. 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 poiché 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 ASP.NET, 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 memorizzate in cache, nelle strutture dati delle istanze delle classi fornite dall'API.

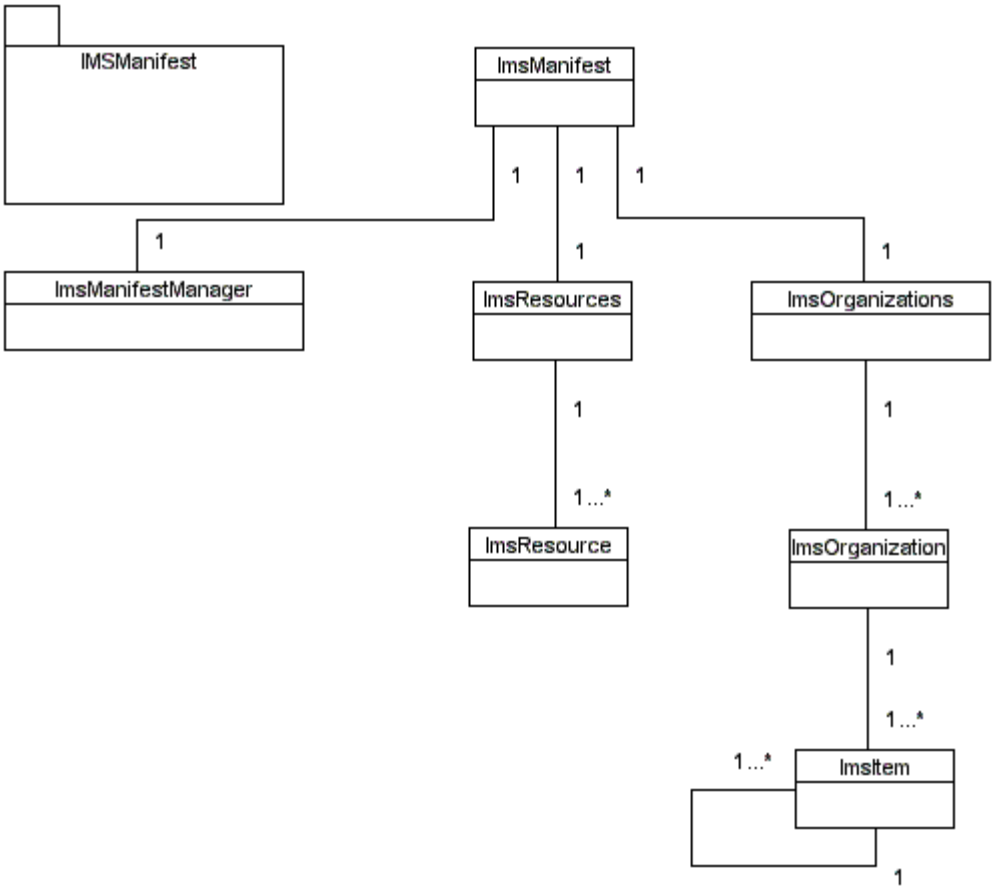


Figura 21: Diagramma delle classi dell'API per la gestione (in sola lettura) del documento XML imsManifest.xml dello standard IMS.

4.3 Architettura

L'architettura funzionale dell'applicazione utilizza come riferimento il modello di LMS proposto da SCORM e visto nel paragrafo 2.2. PocketLezi implementa tale modello conservando tutte le sue funzionalità (anche se, come vedremo, non verrà rispettato pienamente lo standard implementativo a causa delle limitazioni software imposte dal device mobile utilizzato). Dal punto di vista fisico, le diverse unità di servizio sono state distribuite in modo da realizzare una tipica struttura multi tier di cui riportiamo una rappresentazione:

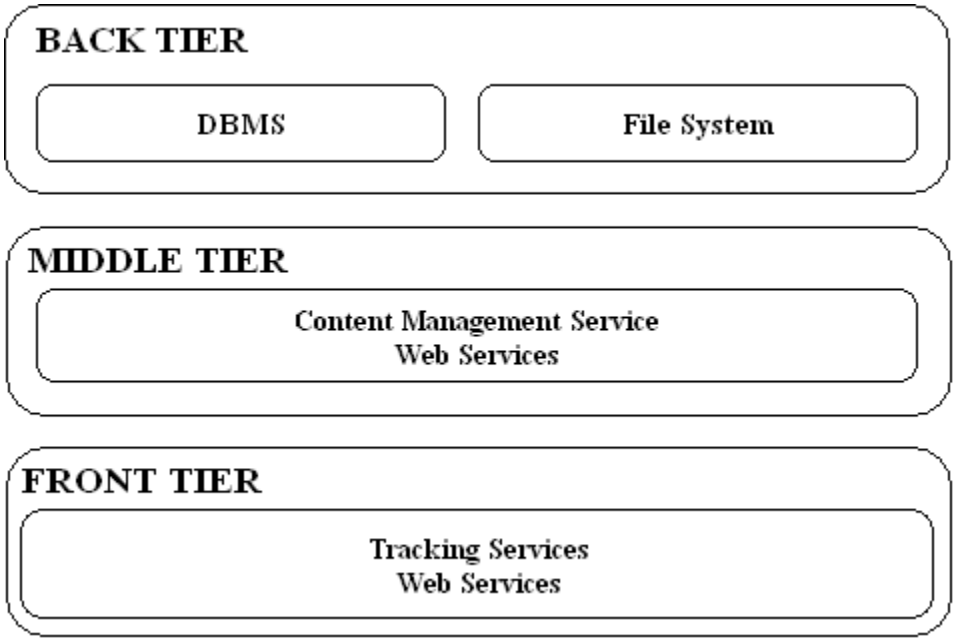


Figura 22: Architettura Fisico-Funzionale di PocketLezi.

1) **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 il salvataggio dei package dei corsi e delle risorse degli utenti utilizzando poi un riferimento contenuto del database per indicizzarli (scelta guidata anche dal desiderio di rimanere fedeli al progetto Lezi.NET). Sarebbe stato possibile anche memorizzare tali file 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 file con dimensioni fino a 3 GB. I file 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 PocketLezi abbiamo adottato una soluzione intermedia. I files delle risorse e i package, come già detto, sono memorizzati su file

system. Nel database è presente un riferimento a tali file ed eventualmente alcuni Metadata. Invece i diversi documenti XML che sono utilizzati dall'applicazione sono memorizzati direttamente nei record insieme agli altri dati. Questi documenti hanno comunque dimensioni di solito inferiori ai 100 KBytes.

2) **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 offerti dal CMS sono resi disponibili al front tier mediante Web Service. 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 Service 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 mediante il browser.

3) **Front Tier**: fornisce servizi che possono essere raggruppati in due insiemi. Il tracking Service, implementato tramite Web Service 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.

La scelta di utilizzare fra il middle tier ed 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 ed una sola porta. Eventualmente in caso di *streaming* sarà necessario aprire altre porte sul firewall. Nelle Figure 23 e 24 è possibile vedere l'architettura fisica di PocketLezi in cui vengono specificate anche le scelte che riguardano la particolare piattaforma software.

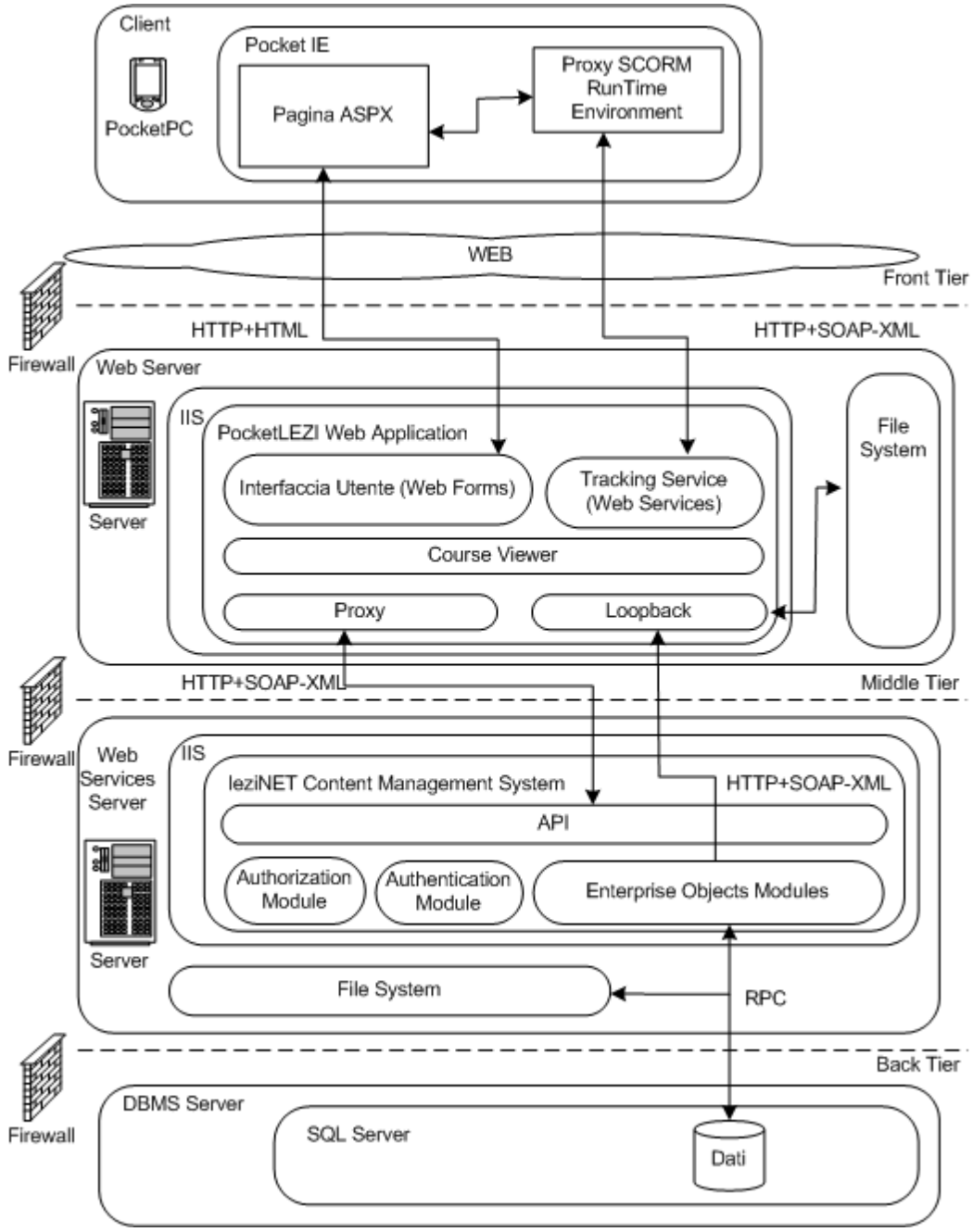


Figura 23: Architettura di PocketLezi con indicazioni di deployment

Nella figura successiva sfruttiamo le specifiche UML per rappresentare al meglio il *deployment* dell'intera applicazione:

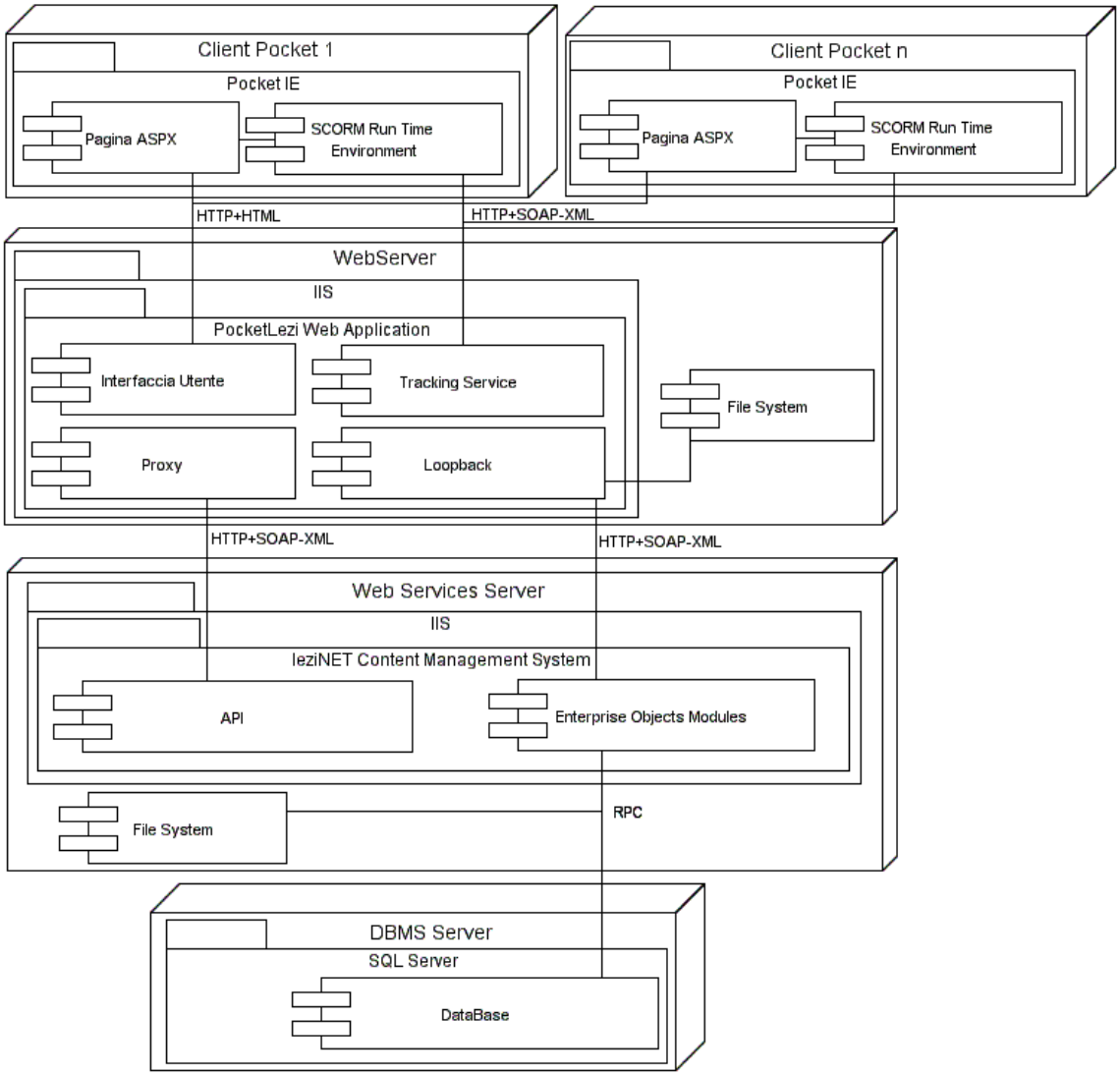


Figura 24: Deployment Diagram dell'applicazione PocketLezi.

4.4 Interfaccia Utente

Nella prima fase di progettazione dell'intera applicazione abbiamo rivolto la nostra attenzione allo studio e realizzazione dell'interfaccia utente. Il primo obiettivo che ci siamo prefissati è stato innanzitutto quello di rimanere il più fedeli possibile all'interfaccia già adottata dall'applicazione Lezi.NET. Questa scelta può essere facilmente spiegata e compresa da motivi di usabilità: un utente già uso alla fruizione dell'ambiente Lezi.NET non riscontrerà alcun problema nell'utilizzo della nostra applicazione in quanto potrà effettuare le operazioni desiderate seguendo gli stessi link,

menu ed icone dell'applicazione a cui è già abituato (ovviamente con le limitazioni imposte dal device utilizzato).
La struttura delle pagine è molto semplice e si poggia essenzialmente su due frameset: uno costituito da un unico frame per sfruttare a pieno l'intera area fruibile dello schermo (seppur sempre ridotta) ed un secondo costituito da due frame orizzontali (quello superiore per i contenuti e/o menu e quello inferiore per visualizzare una sorta di menu per la navigazione).

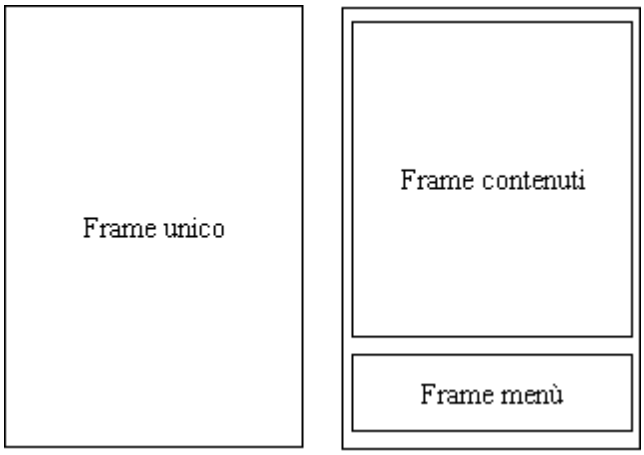


Figura 25: Frameset utilizzati per la strutturazione delle pagine nell'applicazione PocketLezi.

Il fatto che si sia dovuto realizzare solamente l'ambito di fruizione ci ha permesso di semplificare notevolmente la grafica dei vari menu, eliminando le operazioni più "pesanti" dal punto di vista grafico.
Nell'applicazione Lezi.NET la composizione del menu delle azioni che l'utente può eseguire sull'oggetto è determinata 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 e 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.
Nel nostro caso, l'oggetto su cui si possono effettuare delle operazioni sarà sempre e comunque un corso (cosa che elimina la prima operazione di filtraggio) e solamente in ambito di fruizione, ambiente che sarà lo stesso indipendentemente dal ruolo dell'utente

(cosa che elimina la terza operazione di filtraggio). Quindi, in conclusione, le operazioni effettuabili su un corso verranno solamente attivate o meno a seconda dello stato del corso stesso. Dal punto di vista grafico, le diverse azioni, eseguibili sull'oggetto, possono essere efficacemente rappresentate da delle icone munite di etichetta; nel caso in cui una operazione, seppur permessa all'utente, non fosse permessa dallo stato dell'oggetto stesso, l'icona e l'etichetta devono rimanere visibili ma disabilitate.

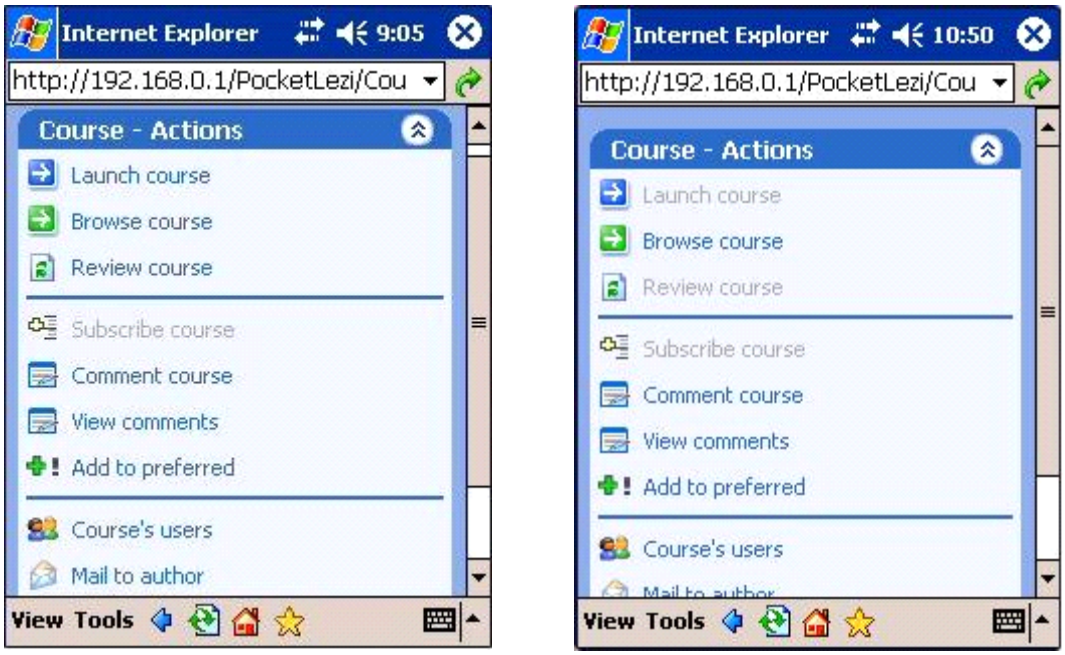


Figura 26: Esempio di come i menu variano a seconda dello stato del corso attualmente in fruizione.

Per quanto riguarda il menu ad albero per la rappresentazione dei contenuti di un corso, si è deciso di usufruire delle librerie grafiche fornite dalla Microsoft come estensione dell'ambiente Visual Studio® .NET. Tale scelta è stata effettuata essenzialmente per due motivi: facilità di utilizzo ed incompatibilità del device con la soluzione adottata in Lezi.NET. Infatti, in quest'ultimo caso, il menu era costruito ed aggiornato in tempo reale tramite uno script Java che andava a scrivere il codice del menu stesso all'interno di un'apposita pagina HTML vuota, inclusa poi nel giusto frame. Questa soluzione presuppone però l'esistenza di più pagine aperte contemporaneamente, cosa proibita da Pocket Internet Explorer, e l'utilizzo pesante di JScript, linguaggio non ancora supportato pienamente dal browser mobile.

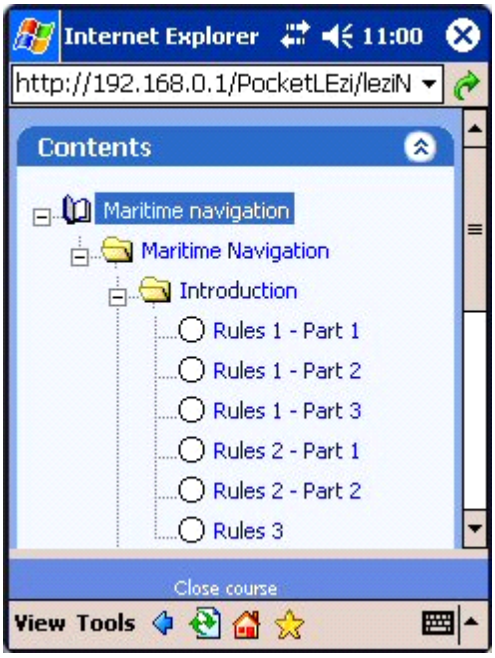


Figura 27: Menu ad albero rappresentante le lezioni contenute in un corso. Tale menu è stato realizzato sfruttando le librerie grafiche della Microsoft (WebControll.dll)

5 Implementazione

5.1 Applicazione PocketLezi

Come visto nel capitolo precedente, l'architettura progettata per l'applicazione PocketLezi porta alla realizzazione di due applicazioni Web: una per tutte le funzionalità del front tier ed una per quelle del middle tier (in questo caso verranno apportate le debite modifiche allo stesso middle tier già utilizzato dall'applicazione Lezi.NET). Per quanto riguarda il back tier, si è usufruito del database già esistente (anche in questo caso apportando le debite modifiche ed aggiunte).

Identificazione dell'utente:
I requisiti del progetto impongono che il sistema possa identificare l'utente che sta seguendo un corso e possa, di conseguenza, regolare l'accesso degli utenti alle varie risorse; per realizzare questo è necessario che l'utente esegua un'autenticazione tramite inserimento di un identificativo utente (userID) ed una password. Questo comporta che se l'utente stesso 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 riconoscimento utente. Il framework ASP.NET fornisce un insieme di classi per poter gestire agevolmente situazioni di questo tipo. Queste classi utilizzate in combinazione con 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 e la pagina Login.aspx.
Riassumiamo i passaggi fondamentali della fase di login con autenticazione Forms:

- 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 è abilitata alla gestione della sessione.

- L'utente esegue la login inserendo userID e password.
- Il sistema verifica le credenziali fornite, rigirando la verifica al leziNETCMS.
- Se la password è corretta, il sistema riporta l'utente alla pagina richiesta in precedenza, in questo caso Default.aspx.
- Dato che 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 ed 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.

La soluzione indicata ha portato i seguenti vantaggi:

- Utilizza un sistema, il sistema Forms, che, insieme ad un set di classi, permette di implementare un solido sistema di autenticazione. Realizzare nuovamente software che fornisca funzionalità già esistenti sarebbe inutile e soprattutto aumenterebbe le linee di codice scritto, aumentando 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.

Riportiamo qui di seguito lo *screenshot* della pagina di login ed il sequence diagram della fase di autenticazione:

Figura 28: Pagina Login.aspx a cui il sistema di autenticazione rimanda per riconoscere l'utente connesso.

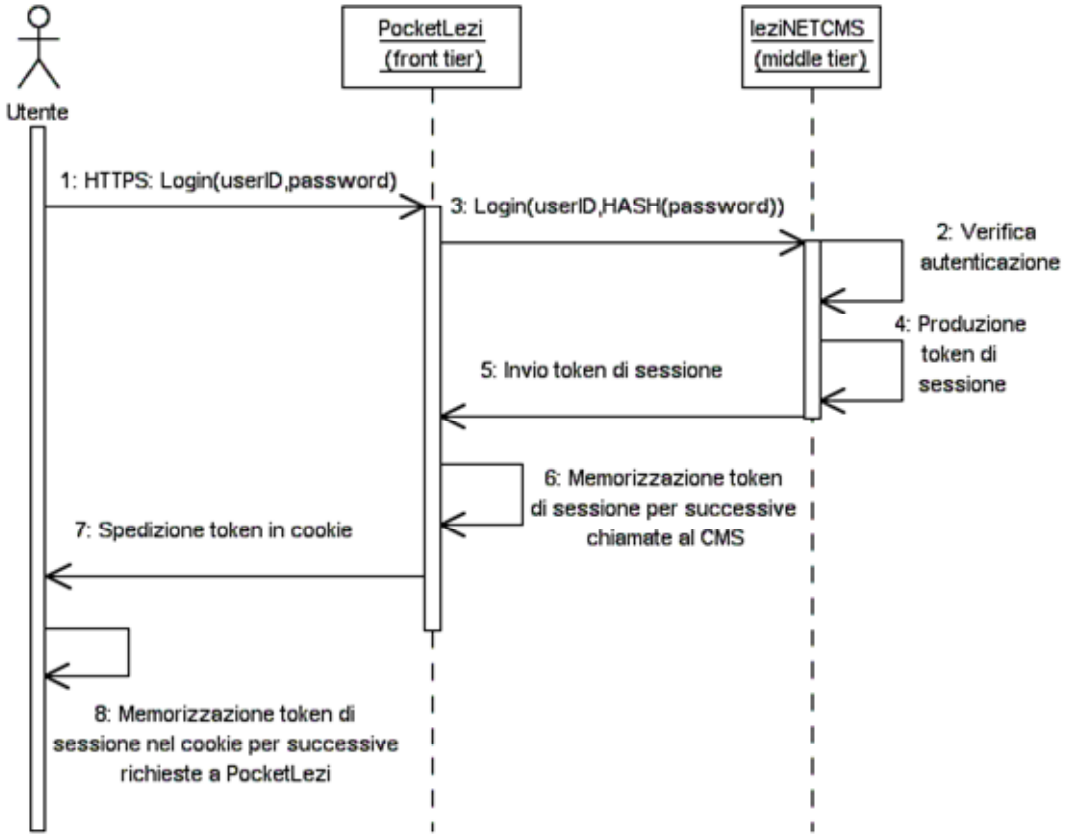


Figura 29: Sequence Diagram della fase di autenticazione.

Accesso e Selezione di un Corso:

Una volta superata la fase di autenticazione l'utente accede alla pagina in cui è possibile vedere l'elenco di corsi disponibili. Ovviamente tale elenco deve includere solo quei corsi che sono effettivamente fruibili da device mobile. Per fare capire al sistema quali corsi rendere disponibili all'utente è bastato fornire l'entità Course di un attributo Type: nel caso in cui il corso sia fruibile da PocketPC tale attributo assumerà valore 1, nel caso in cui il corso sia fruibile da PC fisso o Notebook tale valore sarà pari a 0 oppure non sarà definito. Si è deciso di definire tale attributo come valore numerico anziché come campo boolean per facilitare le successive estensioni del sistema nel campo multimodale (in caso di corso vocale basterà porre tale attributo al valore 2).

Tramite questo attributo è bastato modificare leggermente i comandi di interrogazione dal database per estrarre solamente tutti i corsi con attributo Type pari ad 1. Tali modifiche sono state apportate inoltre a tutti quei metodi collegati ai link di filtraggio sull'elenco dei corsi visibili, come si può osservare dalla figura seguente:

Figura 30: Pagina di visualizzazione dell'elenco dei corsi fruibili da PocketPC

Come si può osservare tale pagina utilizza la struttura costituita da due frame orizzontali: questa scelta è stata fatta per poter rendere sempre visibile (anche nel caso in cui l'elenco dei corsi dovesse causare lo *scrolling* del frame superiore) il link per

poter effettuare il logout dal sistema. Tale operazione di logout comporta il ritorno alla pagina Login.aspx e la distruzione della sessione e di tutti i dati in essa memorizzati.
In caso di selezione di uno dei corsi elencati, il sistema provvede alla memorizzazione nella sessione dell'identificatore del corso scelto e visualizza all'utente la pagina in cui viene fornito il menu con le operazioni effettuabili sul corso stesso. Come già descritto nel capitolo precedente, il sistema provvederà ad abilitare o meno le operazioni effettivamente realizzabili a causa dello stato in cui si trova il corso desiderato.

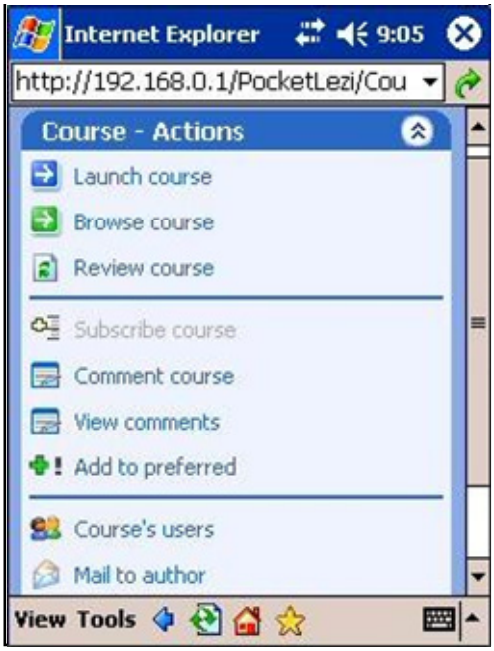


Figura 31: Menu delle operazioni effettuabili sul corso attualmente selezionato. Come si può notare alcune delle operazioni sono disabilitate a causa dello stato in cui il corso si trova.

Proponiamo ora una serie di diagrammi per spiegare meglio alcune operazioni tra quelle che è possibile visionare nella figura precedente.

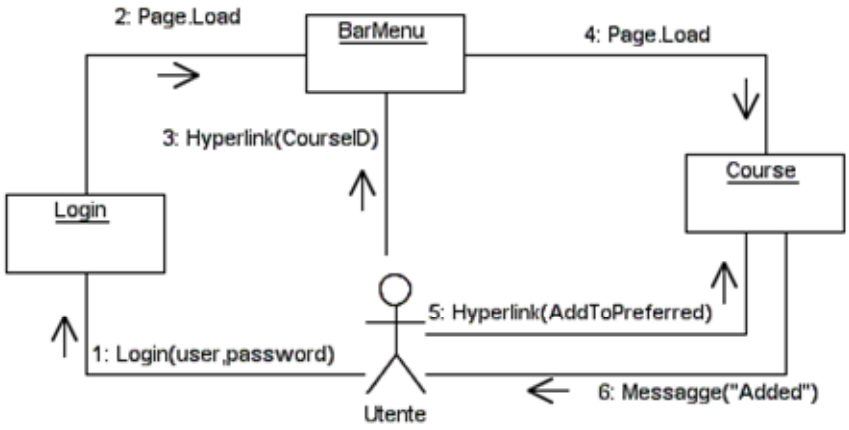


Figura 32: Collaboration Diagram per l'aggiunta di un corso ai propri preferiti.

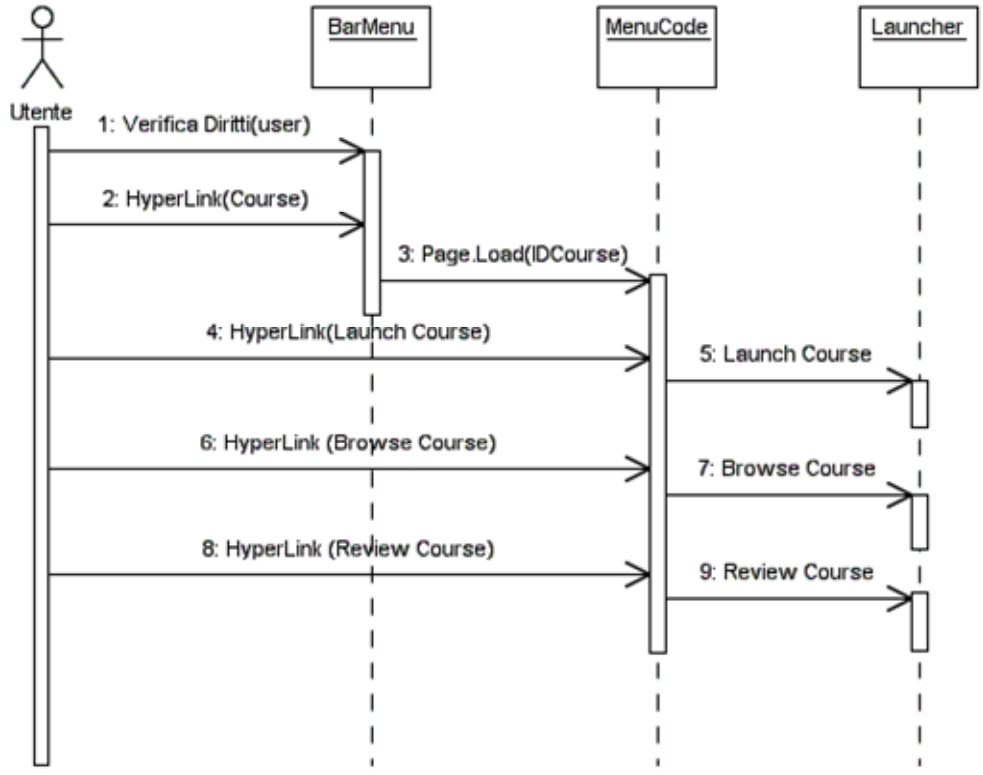


Figura 33: Sequence Diagram rappresentante le tre possibili fruizioni di un corso: la modalità Launch comporta la memorizzazione dell'apprendimento, le modalità Browse e Review invece non tengono traccia delle operazioni che verranno effettuate dall'utente.

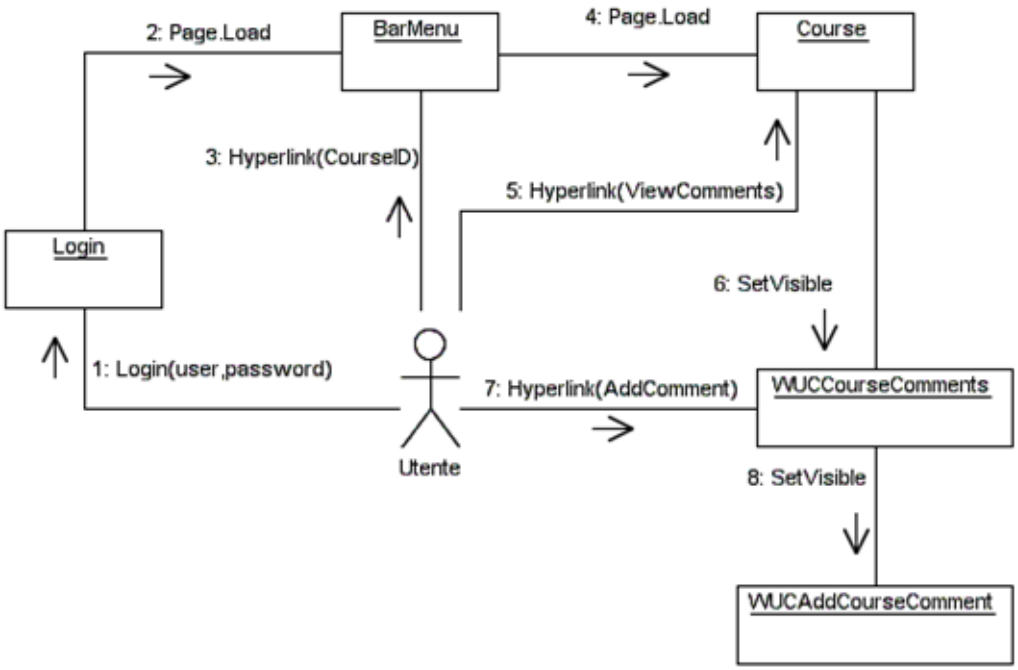


Figura 34: Collaboration Diagram per le operazioni di visualizzazione e/o inserimento di un commento inerente al corso selezionato.

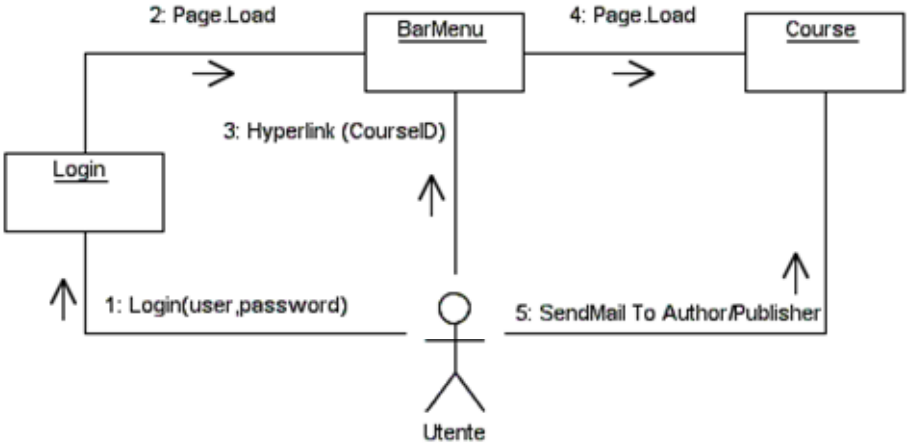


Figura 35: Collaboration Diagram per l'invio di e-mail all'autore o al pubblicatore del corso selezionato.

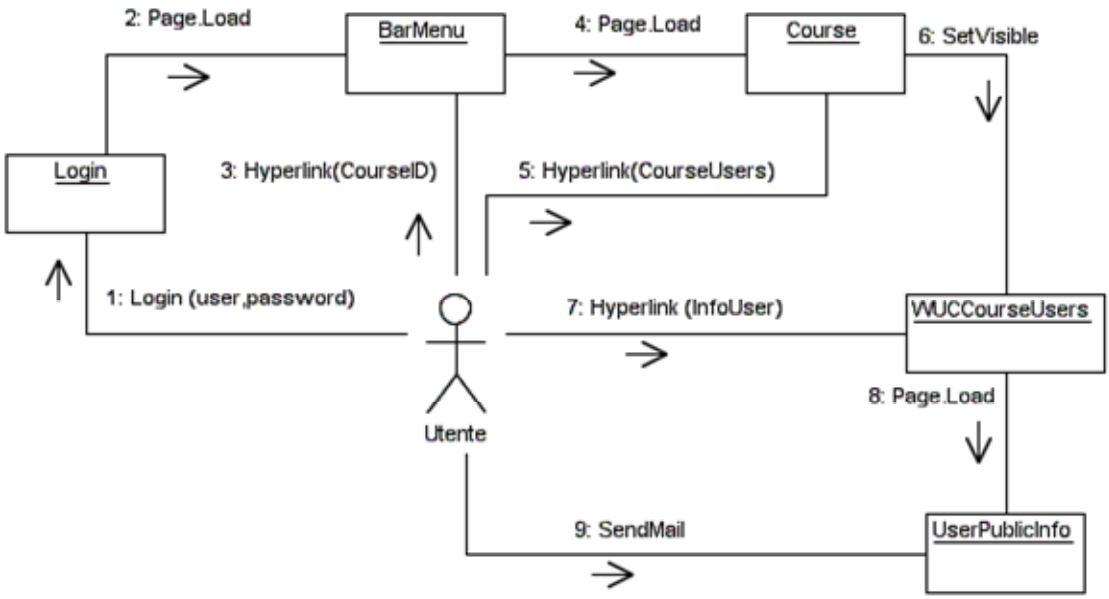


Figura 36: Collaboration Diagram descrivente le operazioni necessarie per poter visualizzare l'elenco degli iscritti al corso e poterli contattare via e-mail.

Fruizione di un Corso:

Approfondiamo ora la fase di fruizione di un corso, fase a cui si accede seguendo il link di Launch di un corso. L'ambiente di visualizzazione dei contenuti deve fornire all'utente gli strumenti per l'interazione con i contenuti e l'ambiente di runtime 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 richiesta che viene spedita al server il quale produrrà una risposta. Questa serie di operazioni causa il caricamento di una nuova pagina nel browser. L'interazione che il run-time ha 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 Web 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 Service, avviene tutta la comunicazione fra il run-time (client side) e l'LMS.

Riassumiamo sinteticamente i passi fondamentali della fruizione di un corso:

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 quest'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 tutti questi aspetti, l'ambiente di fruizione è costituito essenzialmente da due pagine: una contenente l'albero dei contenuti del corso selezionato (MenuCode.aspx) ed una che causerà il "lancio" delle pagine vere e proprie dei contenuti (Launcher.aspx).
La pagina MenuCode.aspx visualizza l'albero dei contenuti sfruttando le potenzialità delle librerie e controlli grafici forniti dalla Microsoft. Per la costruzione di tale albero, si estrapola l'organizzazione del corso dall'imsmanifest.xml. In base a tale elemento si crea ricorsivamente l'albero suddetto: ogni lezione è caratterizzata da una etichetta e da una icona che cambierà a seconda dell'attuale coppia utente-sco (gli stati possibili sono *not attempted*, *incomplete*, *completed*, *passed* e *failed*). Selezionando l'etichetta della lezione voluta si causa il lancio della pagina Launcher.aspx che manda in esecuzione la pagina dei contenuti desiderata.

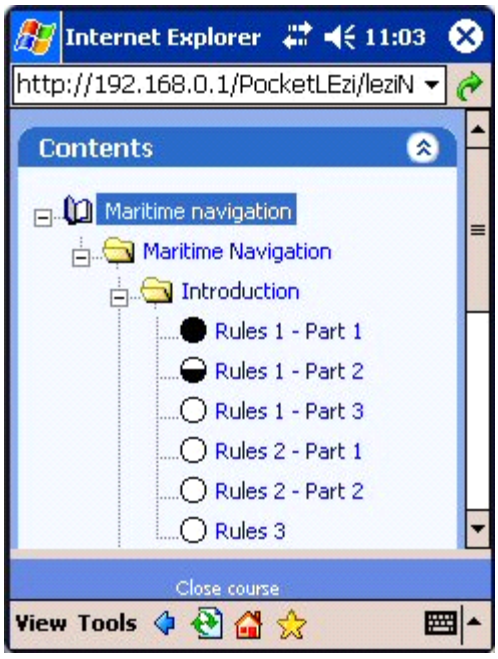


Figura 37: Pagina MenuCode.aspx che visualizza l'albero dei contenuti del corso selezionato. Tramite il link "Close Course" si abbandona l'ambito di fruizione e si torna alla pagina Default.aspx.

Implementazione del Run-Time:

L'implementazione del run-time è stata completamente cambiata rispetto a quella implementata dall'applicazione Lezi.NET. Le specifiche IMS-SCORM impongono all'ambiente di esecuzione di implementare delle API per fornire l'interfaccia necessaria alla comunicazione tra i contenuti SCORM e l'LMS. Poiché il tracking Service dell'LMS in Lezi.NET è stato realizzato tramite Web Service, nella piattaforma originaria, tali interfacce sono costituite da una classe JScript che, tramite un DHTML behaviour, crea, invia e riceve messaggi SOAP-XML verso o dall'LMS.

Tale soluzione risulta inapplicabile attualmente al caso di PocketLezi in quanto il browser Pocket Internet Explorer non fornisce per ora il supporto a tali DHTML behaviour. L'utilizzo di tale oggetto risulta fondamentale in quanto si occupa della costruzione e di tutte le elaborazioni dei pacchetti SOAP-XML dovuti all'utilizzo dei Web Service.

La soluzione a questo problema ci è stata fornita dal fatto che il Sistema Operativo PocketPC 2003 fornisce il pieno supporto al framework .NET. Questa caratteristica ci ha permesso di realizzare l'interfaccia desiderata direttamente dai Web Service che implementano il tracking Service dell'LMS. Infatti si è trattato solamente di creare una

classe proxy C# a partire dai Web Service sopra citati e di invocare direttamente i metodi da tale classe. L'invocazione di tali metodi non avviene più tramite comandi JavaScript ma sarà legata agli eventi di apertura e chiusura della pagina Launcher.aspx.

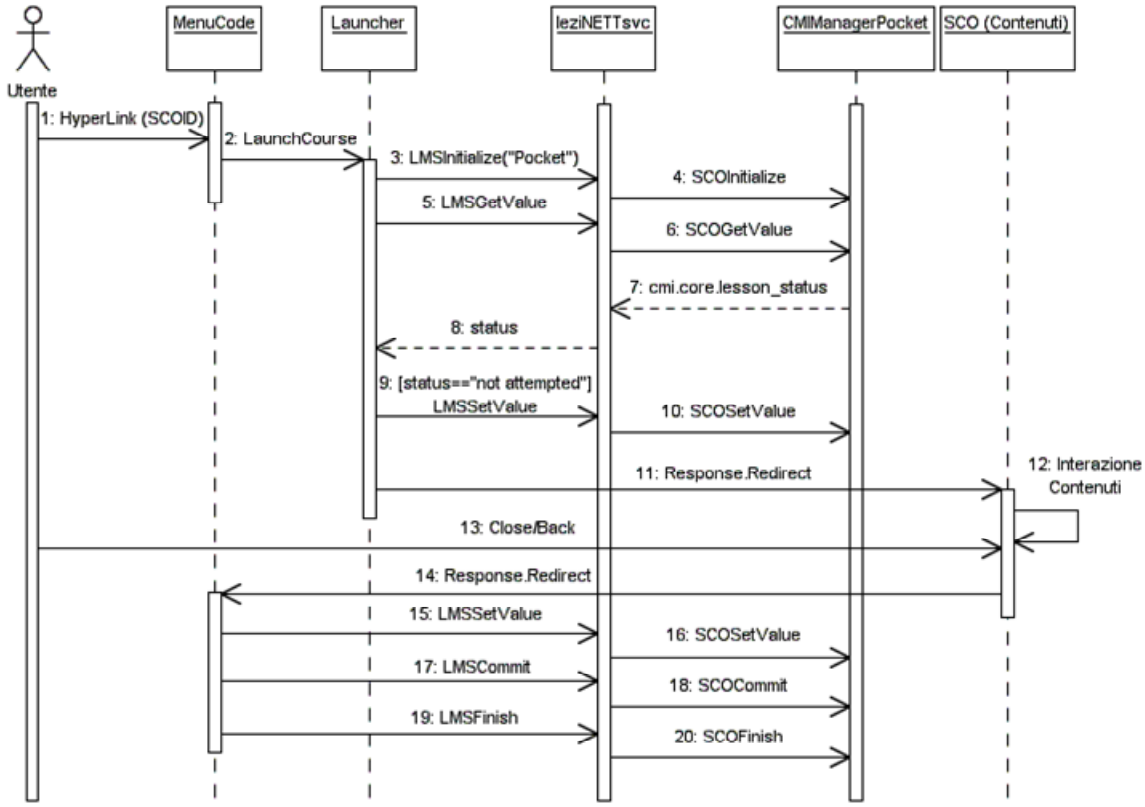


Figura 38: Sequence Diagram rappresentante la fase di fruizione e tracciatura dei contenuti.

5.2 Sincronizzazione dei Contenuti

Con il termine *Sincronizzazione dei Contenuti* intendiamo la capacità del sistema PocketLezi + Lezi.NET di tenere traccia dell'apprendimento di un utente per un corso fornito sia in versione PocketPC che in versione PC Desktop. Come espresso nel capitolo introduttivo, il nostro obiettivo non è quello di creare un ambito di fruizione completamente separato dalla piattaforma già esistente ma quello di fornire gli stessi contenuti in modalità diverse a secondo del mezzo di collegamento utilizzato. Questo fatto comporta delle conseguenze precise: se un utente connesso all'applicazione PocketLezi compie un certo percorso di apprendimento, il sistema deve tenere traccia sia di tale processo che di quello equivalente nella versione Desktop. In tal modo lo

stesso utente, che si volesse connettere in un istante successivo all'applicazione dal PC di casa, non perderebbe il lavoro già svolto su dispositivo mobile e potrebbe continuare il suo processo di apprendimento da dove lo aveva interrotto.

Nella realizzazione dell'ambiente di fruizione abbiamo perciò dovuto tenere conto del fatto che un corso fosse o meno fruibile tramite altri device. Come già esposto nel paragrafo precedente all'entità Course è stato aggiunto un attributo Type necessario per conoscere la "natura" del corso stesso: se Type=1 allora il corso è fatto per essere fruito da PocketPC, se Type=0 o NULL allora il corso è fruibile solamente da PC Desktop o Notebook. Abbiamo poi creato una mappatura tra corsi e tra lezioni per capire quali entità siano connesse da un punto di vista dei contenuti; questa mappatura ha portato alla creazione di due tabelle nel DataBase (*MappaturaCorsi* e *MappaturaSCO*, di cui parleremo nel capitolo 5.4) le cui colonne riportano gli identificatori delle entità logicamente connesse. Tramite queste tabelle sarà sempre possibile sapere quale altro legame Utente-SCO andrà aggiornato oltre ovviamente a quello attualmente in fruizione. Ad esempio, se il corso 110 ed il corso 280 rappresentano lo stesso corso per quanto riguarda i contenuti, con il primo fruibile tramite Desktop ed il secondo da PocketPC, allora avremo nella tabella MappaturaCorsi una riga con elementi 110 e 280. Allo stesso modo se la prima lezione del corso 110 (ad esempio con identificatore 100) porta alla creazione di tre lezioni nel corso 280 (ad esempio con identificatore 200,201 e 202), allora avremo nella tabella MappaturaSCO tre righe con elementi 100-200, 100-201, 100-202. In generale, nel caso di PocketLezi, avremo sempre un rapporto uno a uno tra i corsi, mentre nel caso degli SCO avremo che uno SCO in versione Pocket corrisponderà ad un solo SCO in versione PC, uno SCO in versione PC potrebbe corrispondere invece a più SCO in versione Pocket (questo fatto è dovuto essenzialmente al bisogno di ridurre la quantità di informazioni contenute in una singola pagina fruibile da PocketPC rispetto alla stessa pagina fruibile in versione PC).

Dobbiamo ora fare delle considerazioni inerenti il legame tra stessi contenuti ma in versioni differenti, ovvero lo stesso corso in versione PC ed in versione PocketPC.

1. Dal punto di vista dei contenuti, all'interno delle pagine in versione PC (solamente in quelle in formato HTML testuale e non quelle multimediali) abbiamo dovuto inserire degli elementi (bottoni di domande intermedie per sfruttare la stessa tecnica già implementata dall'applicazione originaria) in modo

da conoscere a che punto verticale della pagina l'utente è arrivato (per risalire a quale lezione in versione Pocket bisogna fare riferimento).

2. Al momento della creazione del file *imsmanifest.xml* del corso in versione PocketPC bisogna tenere presente a quale corso e a quale lezione in versione PC si sia legati dal punto di vista dei contenuti. Questa precisazione è necessaria in quanto al momento della tracciatura dell'apprendimento sarà necessario conoscere una serie di attributi appartenenti agli SCO sia in versione PocketPC che in versione PC. Infatti, oltre ai normali *identifier* dei corsi e degli SCO coinvolti, è necessario ottenere anche *l'orgidentifier* e *l'ididentifier*, che normalmente verrebbero caricati nella sessione al momento della creazione dell'albero dei contenuti. Dato che non è pensabile di caricare "virtualmente" anche l'albero dei contenuti del corrispondente corso PC, la possibilità di legare semanticamente tali attributi alleggerisce la parte implementativa non obbligando l'applicazione a continue interrogazioni sul DataBase. Nell'*imsmanifest.xml* in versione PocketPC è necessario che gli identifier degli SCO siano della forma *sco**n-x*** dove con *sco**n*** si indica l'identifier dello SCO corrispondente in versione PC e con ***x*** il numero progressivo di segmentazione. Es: sco1-2 indica il secondo SCO in versione Pocket derivato dalla frammentazione dello sco1 in versione PC.

   Analogamente bisogna procedere per i *ResIdentifier*, ma in tal caso si avrà *sco_**n**_**x*** con gli stessi significati prima menzionati. Es: sco_1_2 indica il ResIdentifier del secondo SCO in versione Pocket derivato dalla frammentazione dello SCO con ResIdentifer sco_1 in versione PC.
3. Per quanto riguarda le lezioni di valutazione o Test, si è deciso di lasciarle invariate nelle due versioni per evitare problemi di inconsistenza nelle risposte fornite al sistema. In tal caso nella tabella MappaturaSCO ci sarà una corrispondenza uno a uno tra tali lezioni.

Completata questa corrispondenza tra le entità corsi e le entità SCO è stata pensata una politica per rendere effettiva la sincronizzazione. Sinteticamente la politica implementata è la seguente:

1. Se si modifica lo stato di uno SCO da "*not attempted*" ad "*incomplete*" devo fare

i seguenti test:

- Se sono su Pocket dovrò tracciare ad "*incomplete*" anche lo SCO in versione PC ottenuto dalla mappatura.
- Se sono su PC dovrò tracciare ad "*incomplete*" il primo SCO in versione Pocket ottenuto dalla mappatura. In seguito si utilizzano i vari sensori prima menzionati in modo tale da mettere a "*completed*" il primo SCO in versione Pocket della mappatura ed ad "*incomplete*" il secondo SCO in versione Pocket (questo ovviamente collegato all'evento di click sul primo bottone) e così via.

2. Se si modifica lo stato di uno SCO da "*incomplete*" a "*completed*" devo fare i seguenti test:
   - Se sono su Pocket devo testare lo stato degli altri SCO in versione Pocket connessi logicamente allo stesso SCO in versione PC di quello attualmente in fruizione (questo sempre tramite la tabella MappaturaSCO). Se tutti questi altri SCO in versione Pocket hanno lo stato a "*completed*" allora dovrò tracciare a "*completed*" sia lo stato dello SCO in versione Pocket corrente che quello dello SCO in versione PC logicamente connesso ad esso.
   - Se sono su PC dovrò tracciare a "*completed*" anche lo stato di ogni SCO in versione Pocket logicamente connesso allo SCO attuale.

Per rendere più chiaro questo punto chiave riportiamo nella pagina seguente due figure rappresentanti gli State Charts Diagram della politica di Sincronizzazione dei Contenuti lato PocketPC e lato PC.

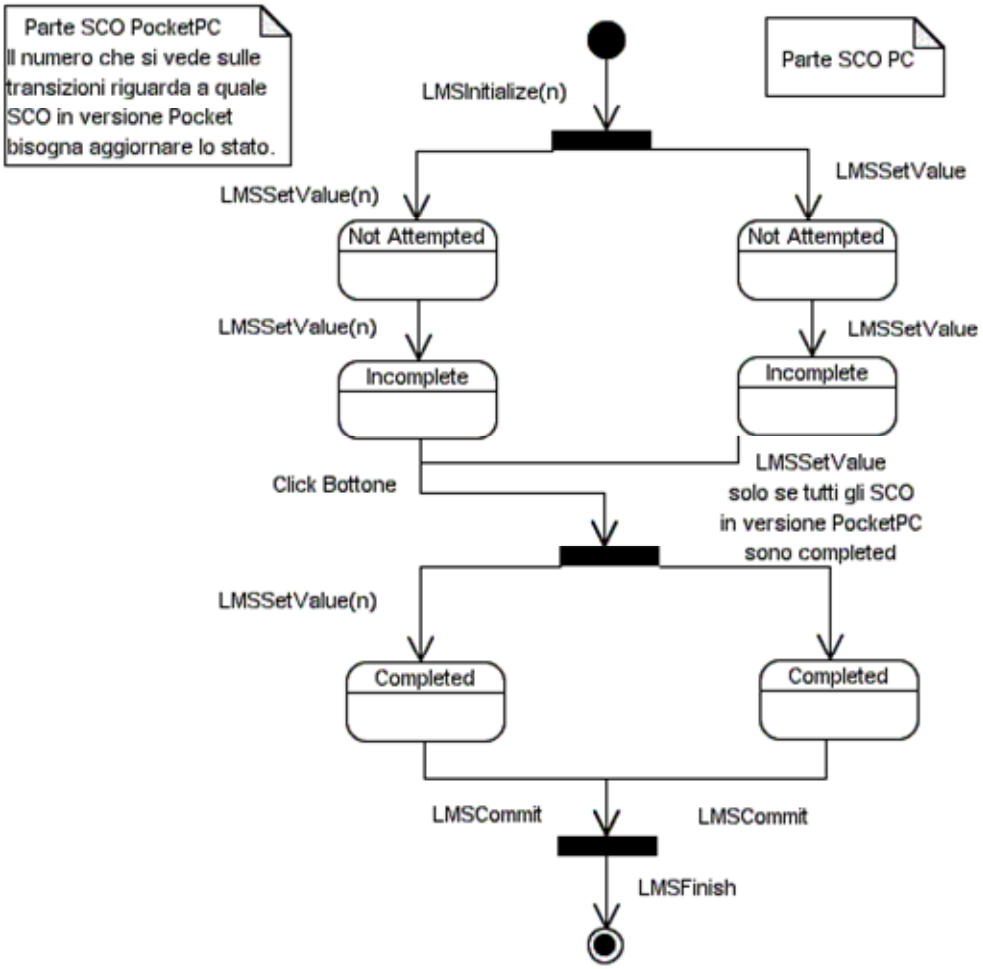


Figura 39: State Chart Diagram della politica di Sincronizzazione lato PocketPC.

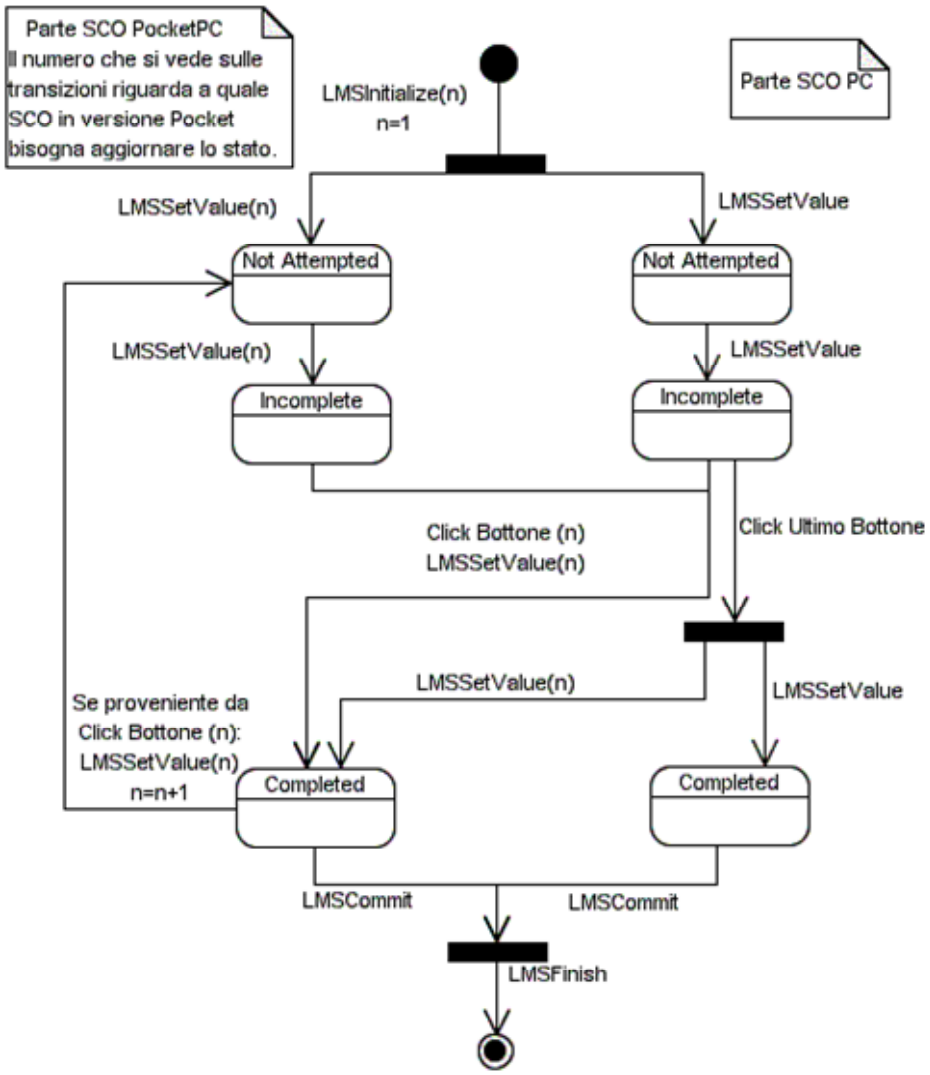


Figura 40: State Chart Diagram della politica di Sincronizzazione lato PC

Poiché la memorizzazione dello stato della coppia Utente-SCO avviene all'interno della struttura dati CMI (vista nel capitolo 2.3.2.4), e poiché questa è accessibile e modificabile solo attraverso la classe CMIManager, la realizzazione vera e propria della sincronizzazione dei contenuti ha comportato la creazione e memorizzazione nella sessione di due Manager contemporanei: uno per occuparsi delle modifiche apportate agli SCO in versione Pocket, l'altro di quelle apportate agli SCO in versione PC.

L'ultimo passo fatto è stato quello di duplicare tutte le chiamate di modifica dello stato all'LMS in modo tale che gli SCO modificassero e memorizzassero il proprio stato, sempre tenendo presente la politica prima menzionata.

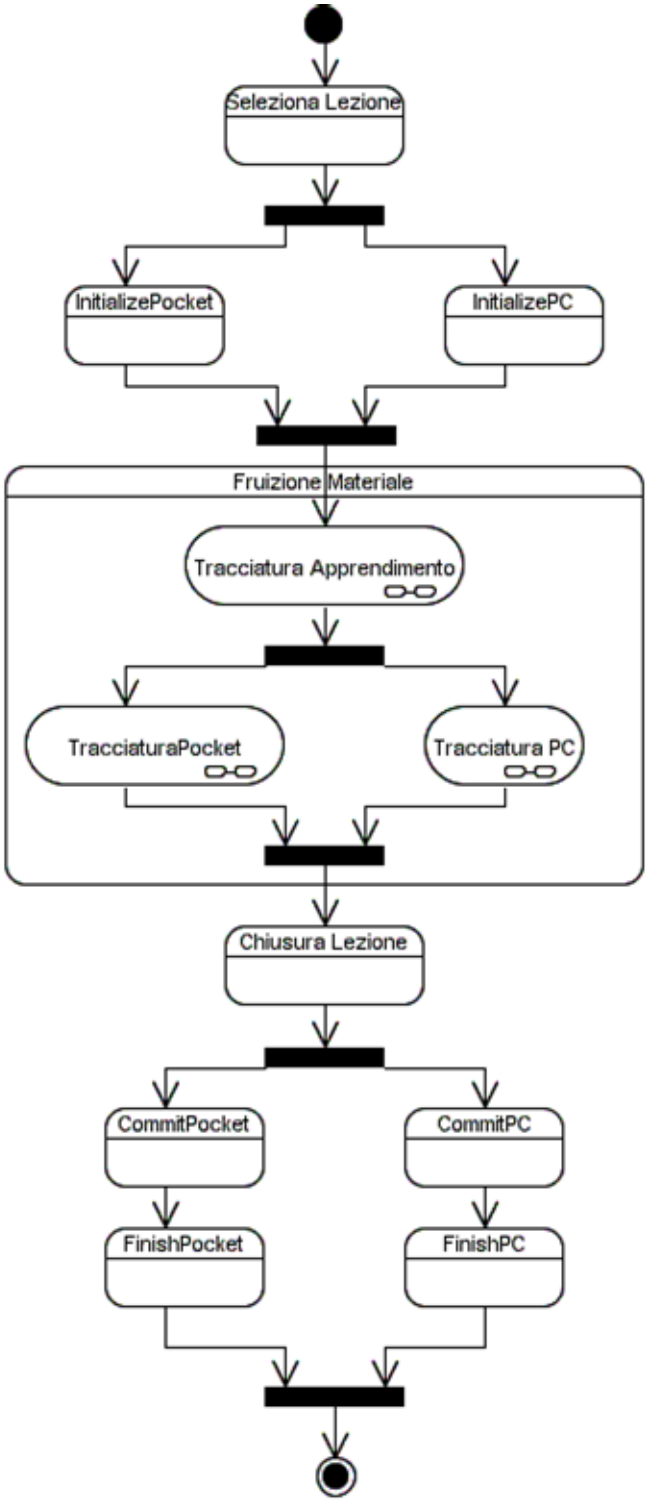


Figura 41: Activity Diagram rappresentante la sincronizzazione dei contenuti durante la fruizione di un corso.

Di seguito riportiamo delle schermate esplicative dell'applicazione PocketLezi e di Lezi.NET per chiarire ancora meglio il problema della sincronizzazione:

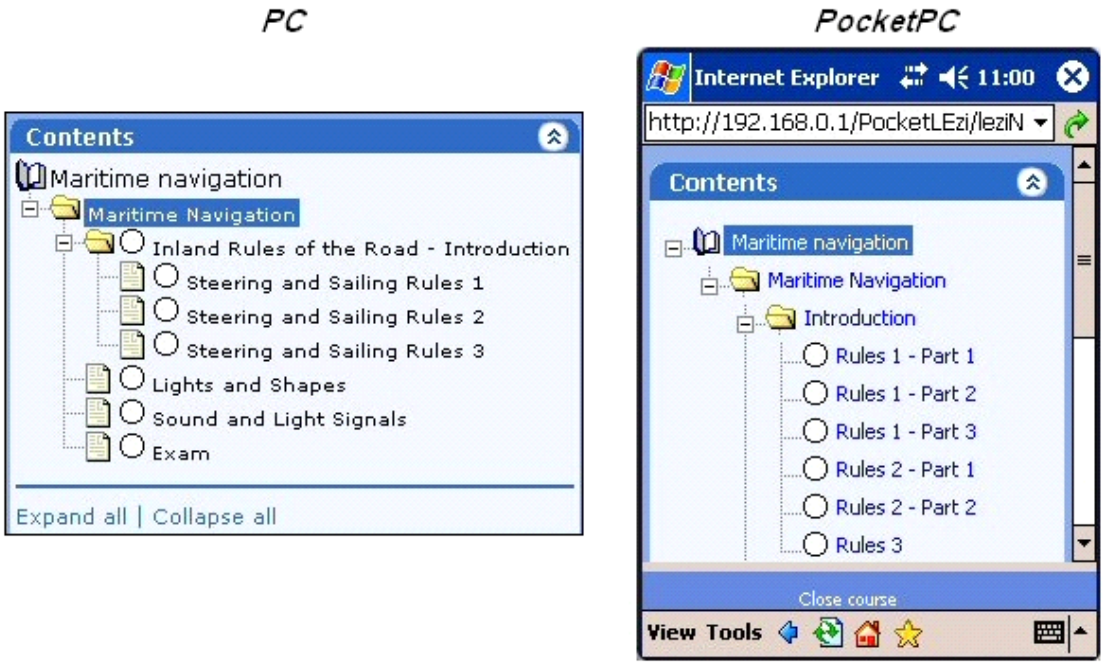


Figura 42: Stato iniziale di una lezione

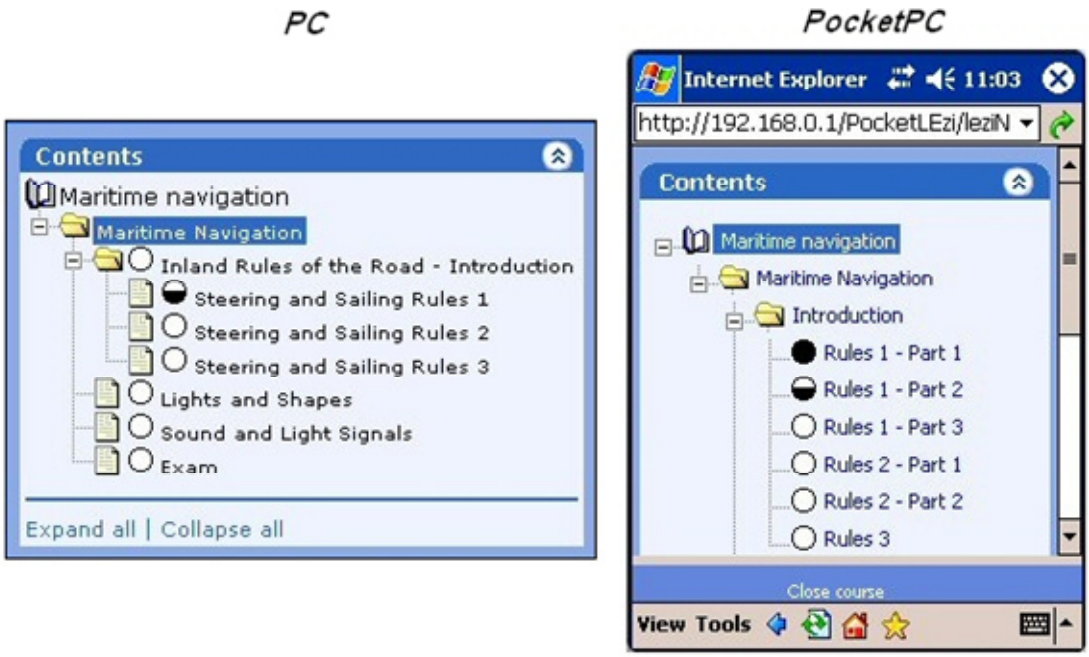


Figura 43: Stato intermedio di una lezione conseguente all'evento di click del primo bottone intermedio di una lezione in versione PC, oppure alla fine della prima lezione in versione PocketPC ed inizio della seconda.

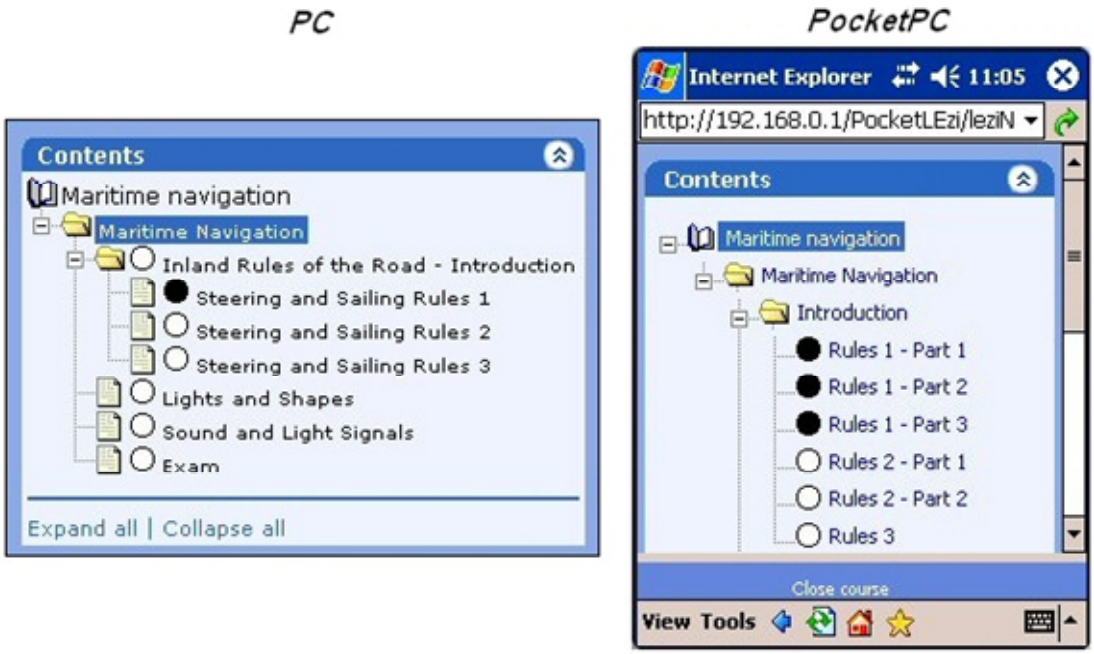


Figura 44: Stato finale della prima lezione lato PC e delle prime tre lato PocketPC. Tali tre lezioni unite sono equivalenti dal punto di vista dei contenuti alla prima lezione in formato PC.

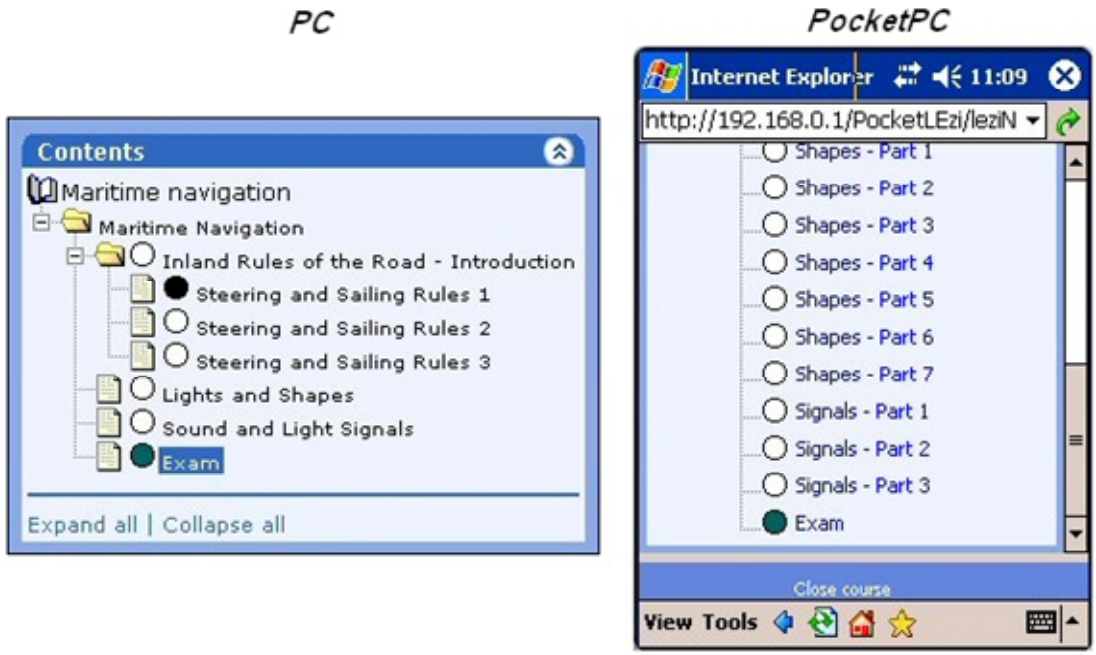


Figura 45: Stato passato di una lezione di Test finale; come si vede in entrambi i dispositivi la corrispondenza è uno a uno. In caso di esame fallito l'icona sarebbe stata di colore rosso.

.

5.3 Applicazione 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 nel capitolo 4.3 , sono implementati mediante Web services pubblicati dal Web server IIS. Il CMS ha la struttura rappresentata nella figura seguente:

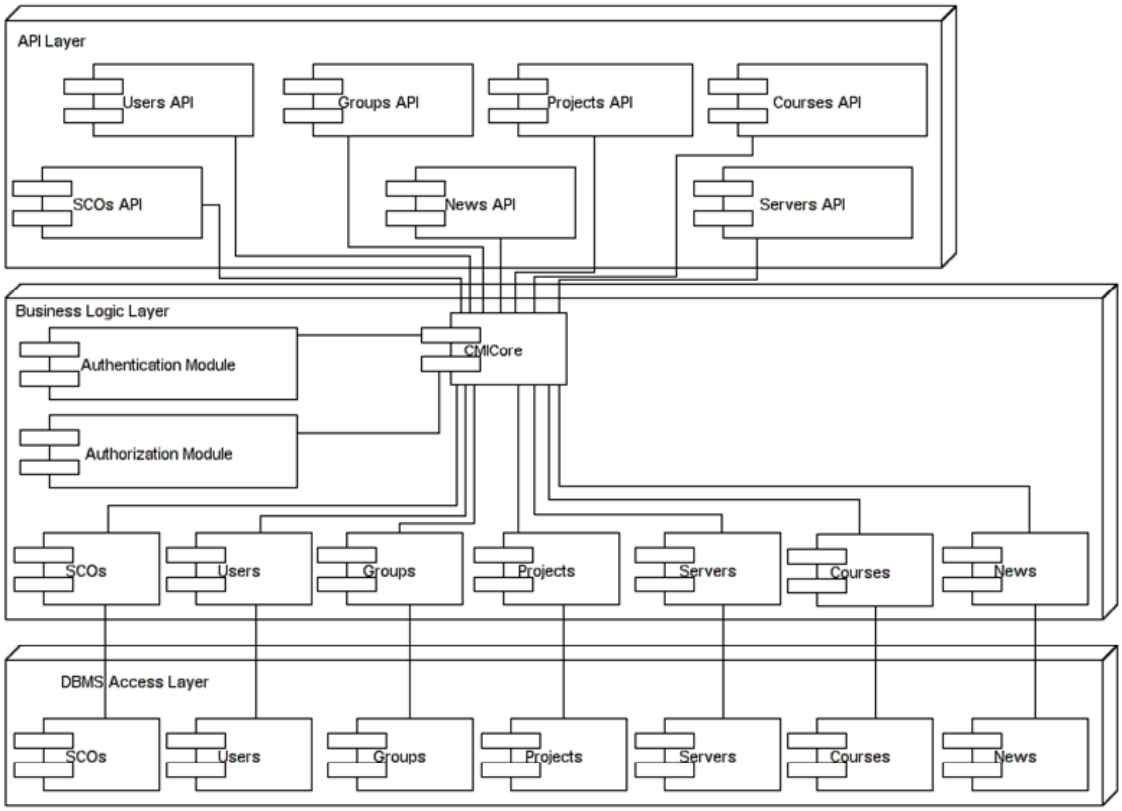


Figura 46: Deployment Diagram dei diversi componenti dell'applicazione lezi.NET CMS.

E' stato suddiviso in tre livelli aventi funzionalità diverse. Il primo *layer* si occupa di fornire l'interfaccia per i diversi servizi. Ad esempio, il componente UsersAPI fornisce un insieme di Web Service che permettono al front tier di effettuare tutte le operazioni possibili con gli utenti.

Il secondo livello del CMS è costituito dal *layer* di *business logic*; in questo layer è raccolta tutta la logica applicativa in modo da rendere tutto il più modulare possibile. Questa scelta fa in modo che una modifica all'interfaccia o al DBMS non comporta un cambiamento di questo secondo livello. I due componenti più importanti del layer di business logic sono il modulo di autenticazione ed il modulo di autorizzazione.

5.3.1 Modulo di Autenticazione

Compito di questo modulo è quello di gestire la fase di autenticazione dell'utente. I diversi front end inviano 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 ed un hash della password. Il CMS cerca la password dell'utente corrispondete alla 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.

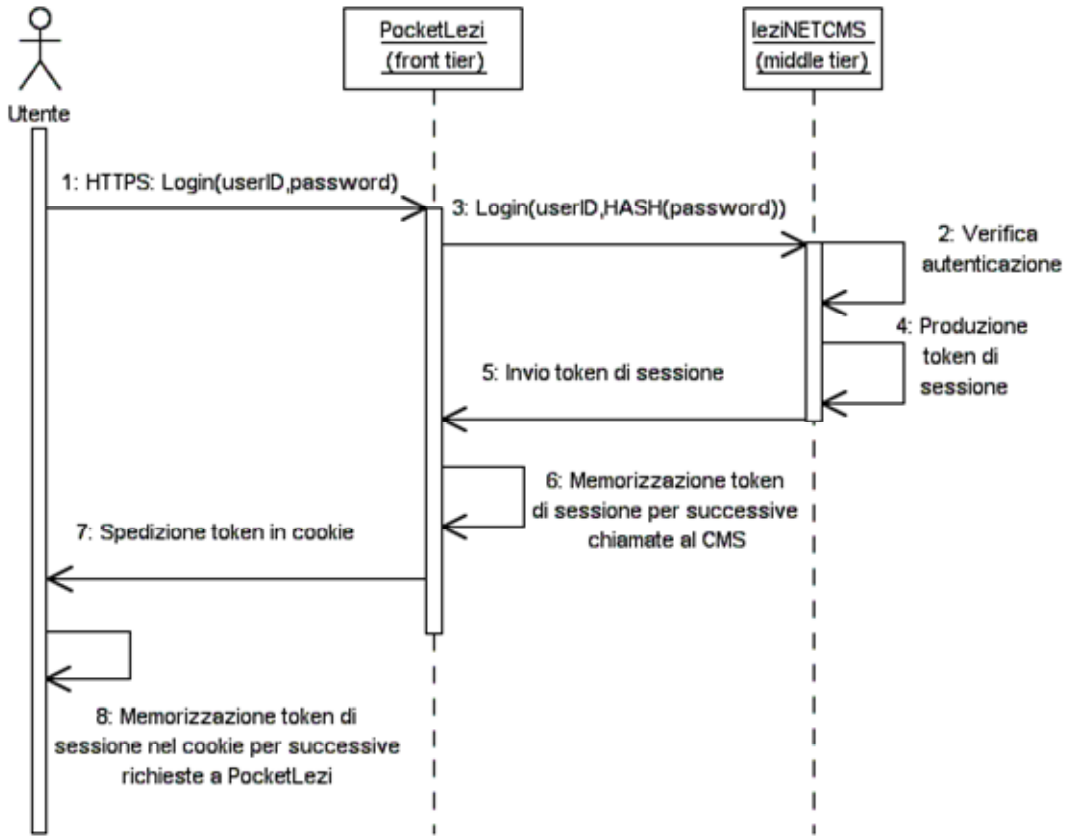


Figura 47: Sequence Diagram della fase di autenticazione di un utente nel sistema.

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 capitolo 5.1) che lo memorizza in un *cookie* sul client. In ogni operazione di *Request* del browser viene spedito 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.

5.3.2 Modulo di Autorizzazione

Dopo che una richiesta ad un servizio del CMS è stata filtrata dal modulo di autenticazione, essa viene elaborata dal modulo di autorizzazione che in base al tipo di richiesta ed al ruolo dell'utente decide se è possibile eseguire l'operazione. 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 di 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 file di log il tentativo di attacco.

5.4 Base di Dati

Il back tier dell'applicazione è composto dalla base di dati che memorizza lo stato del sistema. Parte dei degli schemi presenti in questa sezione appartiene in 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 la lettura di dati mediante *stored procedure*, i record trovati, ad esempio, vengono immediatamente utilizzati per istanziare 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 *Object Oriented*.

Il codice scritto risulta più chiaro e quindi facile da comprendere, modificare e manutenere.

5.4.1 Schema

La figura della prossima pagina presenta lo schema della base di dati utilizzata. Come visto nel capitolo 4.1, 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.
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.
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.
Come espresso nel capitolo 5.2 esistono le due tabelle *MappaturaCorsi* e *MappaturaSCO* necessarie per la realizzazione della sincronizzazione della fruizione dei contenuti sui due canali di accesso al sistema. Queste tabelle non fanno altro che riportare gli identificatori dei corsi e delle lezioni che sono legati tra di loro dal punto di vista dei contenuti.

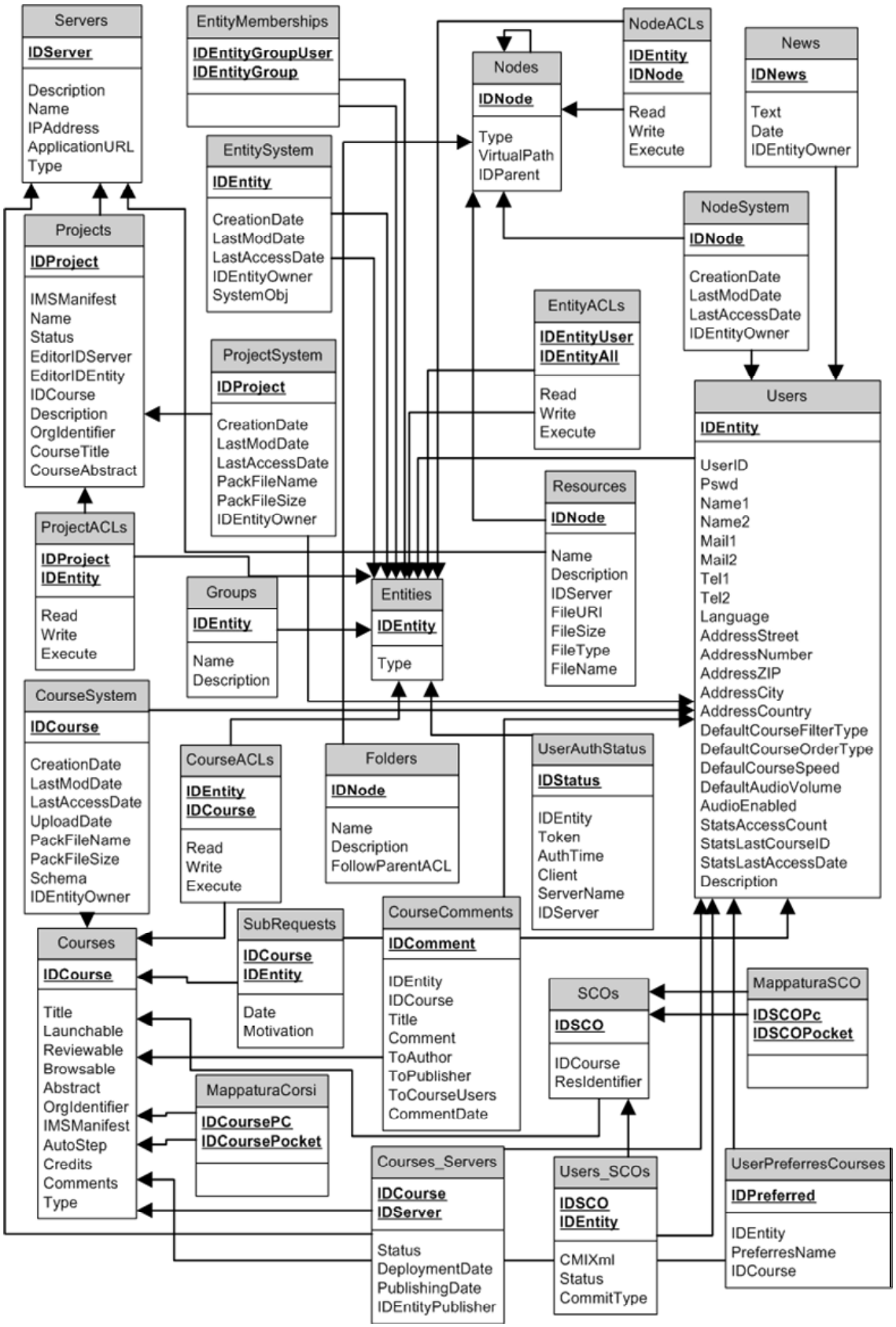


Figura 48: Schema della Base di Dati.

5.4.2 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.
- Buone performance garantite dal fatto che il codice delle stored procedure risiede ed è già pre-compilato presso il DB.
- Desiderio di rimanere il più fedele possibile alle scelte implementative del progetto Lezi.NET.

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 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 seguente è possibile vedere una stored procedure che permette di creare un corso.

Esempio 5: Stored Procedure in Transact-SQL per ottenere tutti gli SCO in versione Pocket che sono in relazione con lo stesso SCO in versione PC

```
CREATE PROCEDURE GetIDSCOFratelli

@resIdentifier nvarchar(255),
@IDCourse int

AS

DECLARE @idscopocket int

SELECT @idscopocket=IDSCO
FROM SCOs
WHERE (IDCourse=@IDCourse) AND (ResIdentifier=@resIdentifier)

DECLARE @idscopc int

SELECT @idscopc=IDSCOPc
FROM MappaturaSCO
```

```
WHERE (IDSCOPocket=@idscopocket)

SELECT IDSCOPocket
FROM MappaturaSCO
WHERE (IDSCOPc=@idscopc)
GO
```

Nel precedente esempio si può notare come i parametri utilizzati delle istruzioni *Select* 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 forniscono inoltre delle classi specifiche per l'utilizzo delle stored procedure e, mediante appositi metodi, permettono di risolvere il problema delle conversioni.

Esempio 6: Codice C# che utilizza la stored procedure vista nell'esempio 5. Vengono utilizzate le librerie ADO.NET per l'accesso al Data Base tramite stored procedure.

```
public ArrayList GetIDSCOFratelli(int IDCourse,
                                  string resIdentifier)
{
      SqlConnection conn = new SqlConnection(m_connectionString);
      SqlCommand cmd = null;

      cmd = new SqlCommand("GetIDSCOFratelli",conn);

      cmd.CommandType = CommandType.StoredProcedure;

      // Parameters
      SqlParameter presIdentifier =
             cmd.Parameters.Add("@resIdentifier",resIdentifier);
      presIdentifier.Direction = ParameterDirection.Input;

      SqlParameter pIDCourse =
             cmd.Parameters.Add("@IDCourse",IDCourse);

     pIDCourse.Direction = ParameterDirection.Input;

      SqlDataReader reader = null;

      try
      {
            conn.Open();
            reader = cmd.ExecuteReader();
            ArrayList al = new ArrayList();
```

```
            while (reader.Read())
            {
                  int lc = 0;
                  string IDSCOPocket =
                                    reader["IDSCOPocket"].ToString();
                  lc=Convert.ToInt32(IDSCOPocket);
                  al.Add(lc);
            }
            return al;
      }

      catch (SqlException sqle)
      {
            Global.LogExceptionOnFile("leziNETCMwsvc:Courses
                        Manager:GetIDSCOFratelli","123",sqle);
            throw;
      }

      catch (InvalidOperationException ioe)
      {
            Global.LogExceptionOnFile("leziNETCMwsvc:Courses
                        Manager:GetIDSCOFratelli","123",ioe);
            throw;
      }

      catch (Exception e)
      {
            Global.LogExceptionOnFile("leziNETCMwsvc:Courses
                        Manager:GetIDSCOFratelli","123",e);
            throw;
      }

      finally
      {
            conn.Close();
      }
}
```

6 Validazione Funzionale

La fase di testing dell'applicazione PocketLezi è stata suddivisa in due parti. Nella prima parte ci siamo concentrati sulla validazione dell'intera piattaforma separatamente da quella di Lezi.NET, con l'obiettivo di testare il corretto funzionamento dell'LMS ed in particolare del tracciamento delle fasi di apprendimento degli utenti. In questa fase ci siamo serviti del corso "Maritime Navigation" così come compare nella versione per PC Desktop senza preoccuparci di come visualizzare al meglio i contenuti su PocketPC.

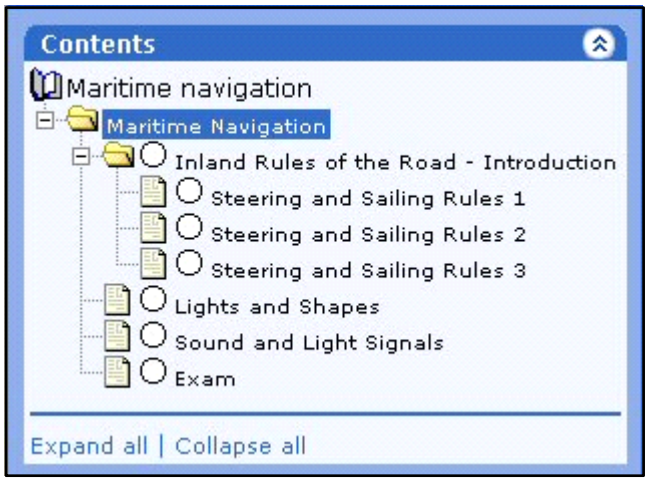


Figura 49: Albero dei contenuti del corso "Maritime Navigation"

Come tipologia di test abbiamo scelto il *Black Box Testing* guidato dagli scenari d'uso; la scelta è caduta principalmente sul Black Box Testing in quanto nel caso di applicazione web come PocketLezi ci è sembrato più significativo utilizzare un approccio di tipo funzionale piuttosto che strutturale, visto inoltre la complessità sia del testing White Box che del codice dell'intera applicazione.

Nei punti considerati cruciali dell'applicazione, quali l'ambito di Run-Time dell'LMS, abbiamo invece deciso di approfondire la fase di verifica utilizzando il *White Box Testing* caratterizzato da *path coverage*: tale scelta è stata guidata dal desiderio di

analizzare a fondo il comportamento del sistema in ogni ramo di controllo dell'ambiente di esecuzione, cercando di prevedere ogni possibile situazione.

A partire dai requisiti espressi nel paragrafo 3.3, sono stati testati i seguenti scenari d'uso:

- Fase di Login sia in caso di utente registrato che di utente non ancora registrato;
- Fase di Logout ovviamente solo in caso di utente registrato e "loggato";
- Visualizzazione dei soli corsi effettivamente fruibili da dispositivo PocketPC con successiva verifica di funzionamento dei filtri applicabili a tale lista di corsi;
- Apertura del menu delle operazioni fruibili su un corso e corretto funzionamento della politica dei diritti di ogni coppia utente-corso. In questa fase abbiamo verificato il supporto dei dispositivi PocketPC per quanto riguarda i fogli di stile CSS;
- Esecuzione degli scenari di invio di posta elettronica ad utenti quali creatore e/o pubblicatore del corso oppure ad altri iscritti al corso attualmente in fruizione;
- Visualizzazione ed invio di commenti inerenti ad un corso;
- Fase di fruizione dei contenuti nelle tre modalità disponibili: Launch, Browse e Review. In particolare ci siamo soffermati sulla modalità Browse approfondendo la validazione con testing di tipo White Box, come accennato in precedenza; tale scelta è stata dettata dal desiderio di verificare tutti i possibili percorsi effettuabili da un utente ed il comportamento prodotto dall'applicazione nel tracciare tale percorsi;
- Chiusura di un corso.

In una fase successiva ci siamo invece soffermati sulla verifica delle operazioni di sincronizzazione tra i due canali di fruizione attualmente disponibili: PC Desktop e PocketPC. Per fare questo è stato necessario utilizzare un corso fornito in due versioni, identiche dal punto di vista dei contenuti ma strutturalmente differenti. La scelta è caduta sempre sul corso "Maritime Navigation".

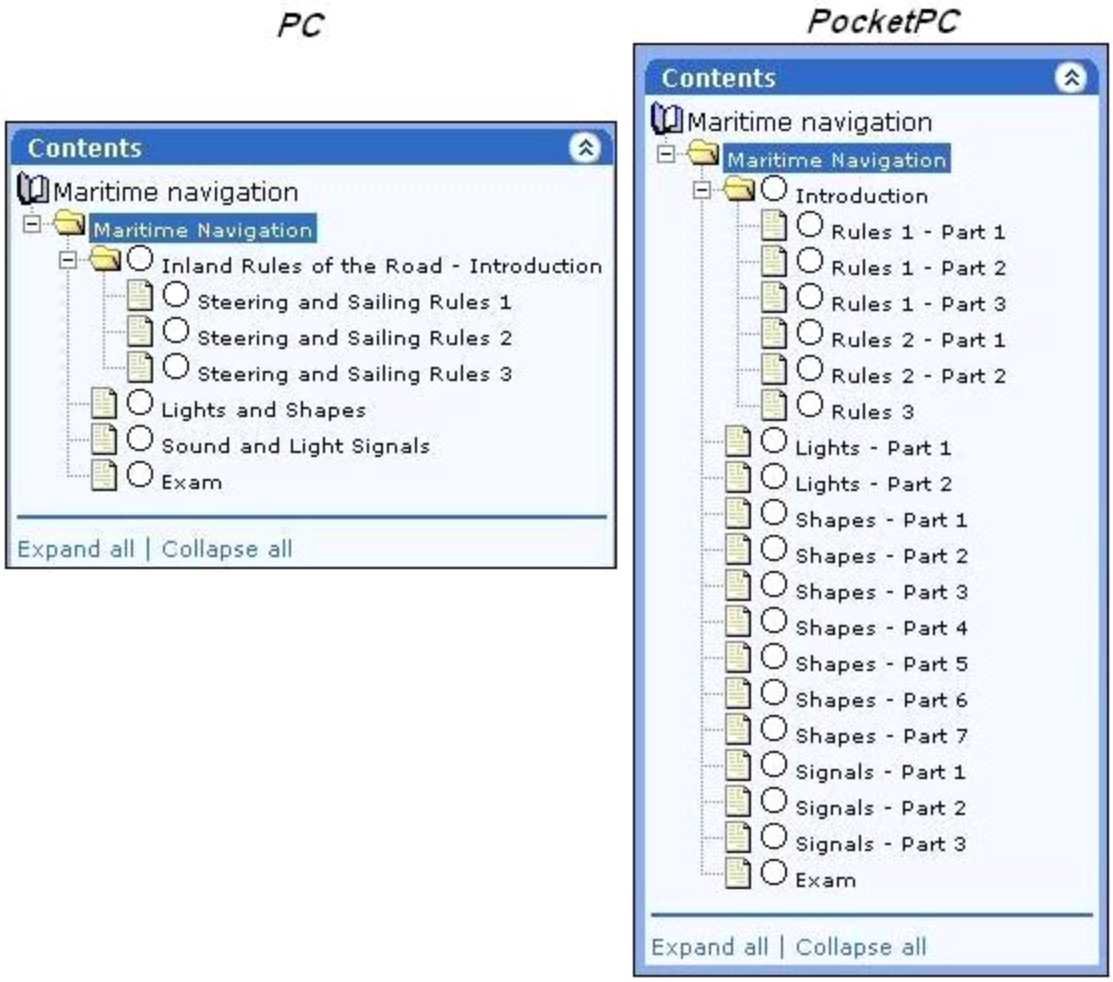


Figura 50: Albero del corso "Maritime Navigation" fornito nelle due versioni per PC e PocketPC

La versione fruibile da PocketPC ha subito le seguenti modifiche a partire dalla corrispondente versione per PC Desktop:

- Ogni pagina di contenuti HTML è stata frammentata in più pagine per tenere conto della ridotta area di visualizzazione disponibile sul nuovo device;
- Alla fine di ognuna di tali pagine sono stati aggiunti:
  - Un bottone per poter identificare la conclusione della fruizione della corrispondente lezione;
  - Un link "Back" per permettere all'utente di tornare al menu ad albero dei contenuti senza terminare forzatamente la lezione;
- Modifica dell'imsmanifest corrispondente:

- Modifica della sezione di *organization* e di *resource* corrispondente alla frammentazione prima menzionata; in questa fase di modifica abbiamo tenuto presente le considerazioni fatte nel paragrafo 5.2 inerenti al valore da attribuire agli identificatori *orgidentifier* e *residentifier*;
- Contrazione dei titoli delle lezioni a seguito sempre della ridotta area di lavoro disponibile.

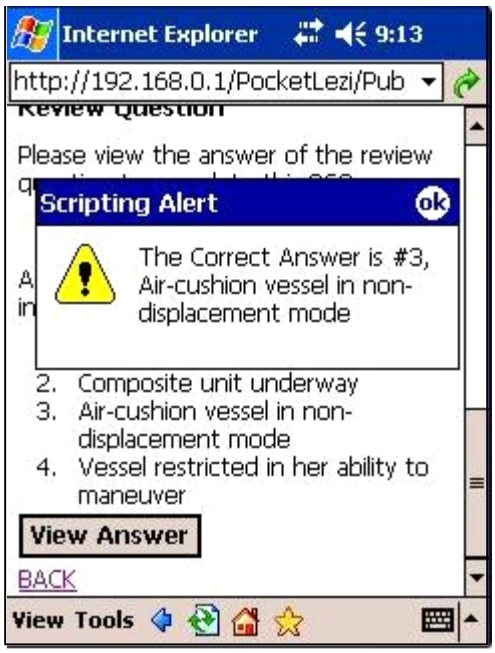
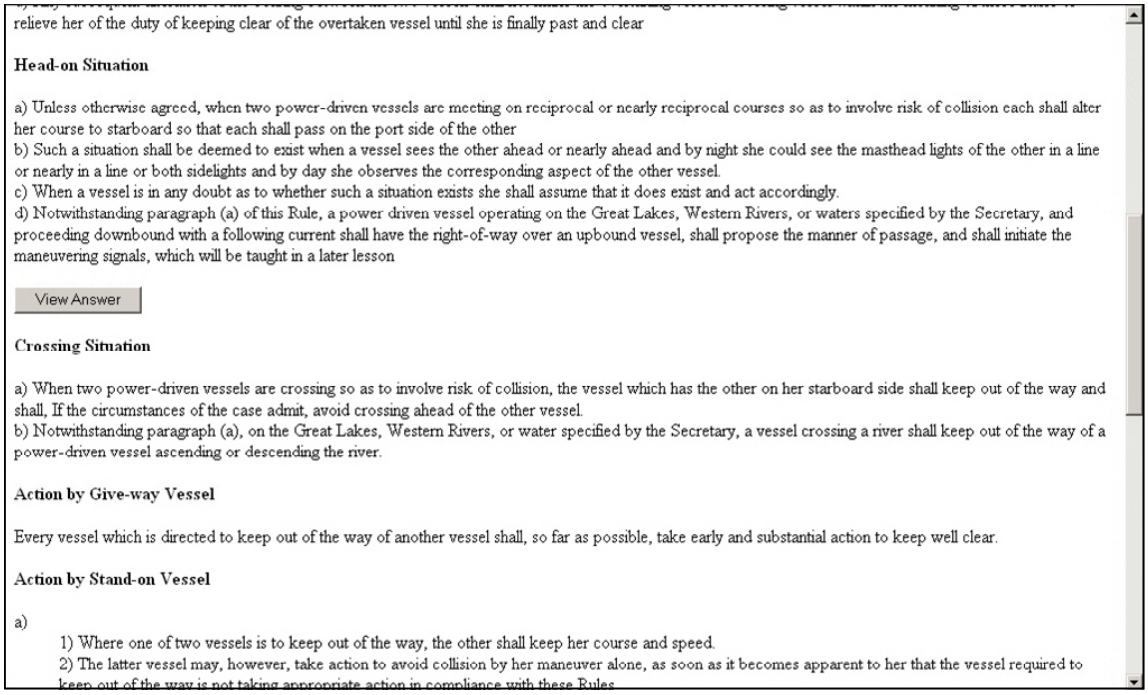


Figura 51: Esempio di nuova pagina di una lezione del corso "Maritime Navigation" per PocketPC

La realizzazione della sincronizzazione ha reso necessarie delle modifiche anche al corso fruibile da PC Desktop; abbiamo aggiunto ad ogni pagina HTML corrispondente ad una lezione dei bottoni intermedi nei punti in cui si è deciso di frammentare il corso per ottenere la versione fruibile da PocketPC. Tali bottoni sono stati collegati ad un gestore di eventi di pressione differente rispetto a quello collegato al bottone conclusivo della lezione: infatti la pressione di un bottone intermedio comporta il cambiamento a *completed* dello stato dello SCO in versione PocketPC corrispondente e l'inizializzazione dello SCO in versione PocketPC successivo, invece la pressione dell'ultimo bottone comporta il solo cambiamento a *completed* dello stato dello SCO appena concluso (sia in versione PocketPC che in versione PC Desktop).


relieve her of the duty of keeping clear of the overtaken vessel until she is finally past and clear

Head-on Situation

a) Unless otherwise agreed, when two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other
b) Such a situation shall be deemed to exist when a vessel sees the other ahead or nearly ahead and by night she could see the masthead lights of the other in a line or nearly in a line or both sidelights and by day she observes the corresponding aspect of the other vessel.
c) When a vessel is in any doubt as to whether such a situation exists she shall assume that it does exist and act accordingly.
d) Notwithstanding paragraph (a) of this Rule, a power driven vessel operating on the Great Lakes, Western Rivers, or waters specified by the Secretary, and proceeding downbound with a following current shall have the right-of-way over an upbound vessel, shall propose the manner of passage, and shall initiate the maneuvering signals, which will be taught in a later lesson

View Answer

Crossing Situation

a) When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her starboard side shall keep out of the way and shall, If the circumstances of the case admit, avoid crossing ahead of the other vessel.
b) Notwithstanding paragraph (a), on the Great Lakes, Western Rivers, or water specified by the Secretary, a vessel crossing a river shall keep out of the way of a power-driven vessel ascending or descending the river.

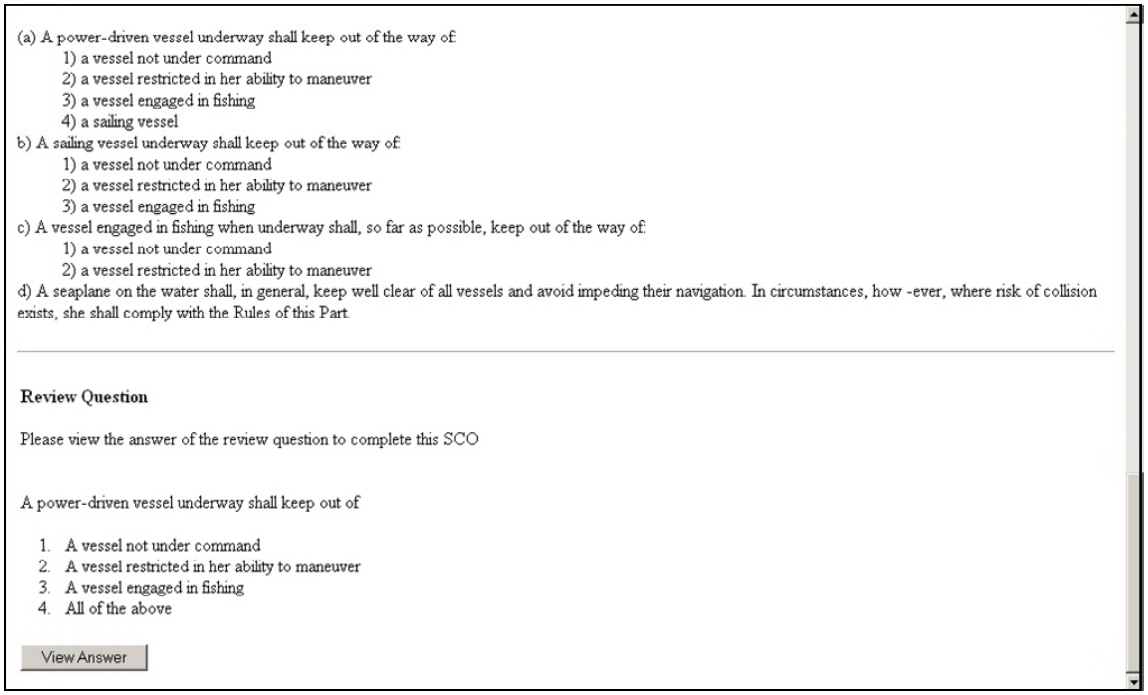
Action by Give-way Vessel

Every vessel which is directed to keep out of the way of another vessel shall, so far as possible, take early and substantial action to keep well clear.

Action by Stand-on Vessel

a)
1) Where one of two vessels is to keep out of the way, the other shall keep her course and speed.
2) The latter vessel may, however, take action to avoid collision by her maneuver alone, as soon as it becomes apparent to her that the vessel required to keep out of the way is not taking appropriate action in compliance with these Rules

Figura 52: Esempio di bottone intermedio inserito nelle lezioni in versione PC Desktop


(a) A power-driven vessel underway shall keep out of the way of:
1) a vessel not under command
2) a vessel restricted in her ability to maneuver
3) a vessel engaged in fishing
4) a sailing vessel
b) A sailing vessel underway shall keep out of the way of:
1) a vessel not under command
2) a vessel restricted in her ability to maneuver
3) a vessel engaged in fishing
c) A vessel engaged in fishing when underway shall, so far as possible, keep out of the way of:
1) a vessel not under command
2) a vessel restricted in her ability to maneuver
d) A seaplane on the water shall, in general, keep well clear of all vessels and avoid impeding their navigation. In circumstances, how -ever, where risk of collision exists, she shall comply with the Rules of this Part.

Review Question

Please view the answer of the review question to complete this SCO

A power-driven vessel underway shall keep out of

1. A vessel not under command
2. A vessel restricted in her ability to maneuver
3. A vessel engaged in fishing
4. All of the above

View Answer

Figura 53: Esempio di Bottone conclusivo di una lezione in versione PC Desktop

Una volta create e modificate tali versioni dello stesso corso abbiamo eseguito la popolazione delle tabelle di mappatura citate nel paragrafo 5.2 per rendere realmente effettiva la sincronizzazione.
Terminata tale fase di creazione è stata ripetuta la fase di *Black Box Testing* unita ad una *White Box Testing* con *path coverage* che ha caratterizzato la parte di verifica della fase di fruizione e tracciamento dei contenuti. La scelta di tali criteri di validazione è stata guidata dalle stesse motivazioni espresse all'inizio del capitolo. Alcune immagini inerenti tale fase di test sono visibili a partire dalla figura 42 di pagina 89.
Durante tale fase di validazione ci siamo soffermati sui possibili scenari d'uso che si possono verificare e ne abbiamo valutato la consistenza ed il corretto funzionamento nel tracciare tali percorsi su entrambi i canali di fruizione attualmente disponibili.

Una sezione a parte va dedicata al testing dei corsi multimediali fruibili da dispositivo PocketPC, ovvero quei corsi i cui contenuti non sono semplici pagine HTML testuali ma filmati WMV (*Windows Media Video*) inclusi all'interno di pagine HTML. Questa parte di verifica non è incentrata né alla verifica del tracciamento delle fasi di apprendimento né del funzionamento dell'LMS in quanto tale testing è già stato effettuato scrupolosamente in precedenza, come prima menzionato.
Per poter effettuare l'inclusione di un filmato all'interno di una pagina web è stato innanzitutto necessario installare sul dispositivo PocketPC i controlli *ActiveX* per poter riconoscere i *plug-in* del lettore multimediale *Windows® Media Player 9 Series*. Una volta installati questi componenti si è proceduto alla conversione dei filmati prescelti nel formato adatto:

- Per poter effettuare lo streaming di un video i file dello stesso devono essere salvati e resi disponibili nel formato WMV;
- Per poter visualizzare tale video su dispositivo PocketPC è necessario effettuare delle operazioni di *encoding* dedicate a tale dispositivo.

Tale fasi sono state eseguite tramite l'utilizzo del programma *Windows® Media Encoder 9* liberamente scaricabile dal sito Microsoft. Una volta creati i filmati nel formato ed *encoding* corretti si è proceduto alla creazione del corso vero e proprio. Il punto focale di tale fase di creazione è stata l'inclusione e funzionamento del filmato

all'interno della pagina statica HTML. L'inclusione è avvenuta tramite oggetto *ActiveX* i cui parametri sono settati da codice *JavaScript*; quest'ultima scelta è stata forzata dalla compatibilità del dispositivo PocketPC con i controlli del lettore multimediale.

Esempio 7: Codice per l'inclusione del lettore multimediale in una pagina HTML.

```
<OBJECT id=PlayerOCX type=application/x-oleobject
    standby="Loading Microsoft Windows Media Player components..."
    height=240 width=208
    classid=CLSID:22D6F312-B0F6-11D0-94AB-0080C74C7E95 VIEWASTEXT>
</OBJECT>
```

Esempio 8: Codice JavaScript per impostare i parametri del controllo ActiveX

```
function start()
{
  PlayerOCX.filename=
      "http://sandro/PocketLezi/PubCourses/Course281/Media.asx";
}
```

Esempio 9: Codice dell'elemento ASX che seleziona il filmato da visualizzare

```
<asx version="3.0">
  <title>Video 2</title>
  <entry>
    <ref href="http://sandro/PocketLezi/PubCourses/Course281/1.wmv" />
  </entry>
</asx>
```

Inserite tali porzioni di codice all'interno della pagina HTML è stato possibile visualizzare correttamente il filmato voluto.

Conclusa questa verifica sono stati aggiunti due bottoni al di sotto del lettor multimediale per poter catturare gli eventi di conclusione e/o interruzione della fase di apprendimento dei contenuti. A seconda del bottone premuto, l'utente porrà fine alla visione del filmato salvando nel database lo stato incompleto, in caso di pressione del bottone "Back", lo stato completo, in caso di pressione del bottone "Complete".

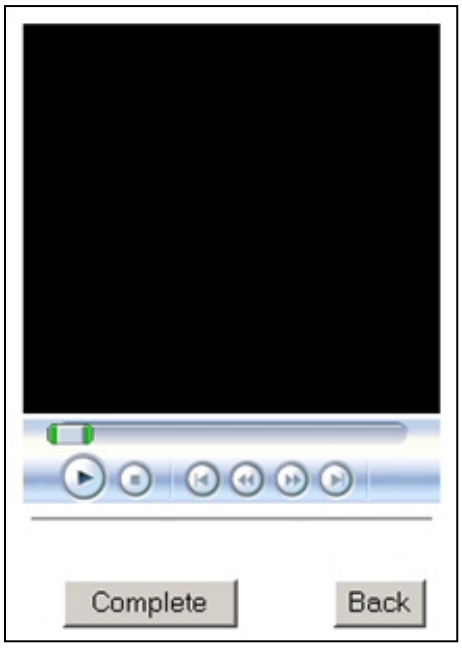


Figura 54: Esempio di Corso Multimediale.

Infine è stata ripetuta la fase di *Black Box Testing* unita ad una *White Box Testing* con *path coverage* che ha caratterizzato la parte di verifica della fase di fruizione e tracciamento dei contenuti. La scelta di tali criteri di validazione è stata guidata dalle stesse motivazioni espresse all'inizio del capitolo.

Si possono prevedere una serie di test futuri non inerenti così strettamente la validazione delle funzionalità dell'applicazione, ma inseriti in scenari d'uso specifici del progetto MAIS. In particolare si testerà l'efficacia di PocketLezi nel suo utilizzo da parte di diversi tipologie di utente (in particolare si valuteranno le sue potenzialità nell'utilizzo da parte di utenti con problemi di apprendimento). Per ognuna di tale categorie di utenti, si analizzeranno a fondo la sua usabilità e l'efficacia della sincronizzazione all'interno dell'intera evoluzione del progetto: PocketLezi + MultiLezi + VocalLezi.

7 Conclusioni

Per facilitare la lettura del capitolo, si riportano i requisiti prefissati dall'applicazione PocketLezi:

- **Ambiente di fruizione per Mobile Device**: si vuole che un qualsiasi utente, fornito ovviamente di PDA, possa connettersi via web all'applicazione e possa così fruire dei contenuti di un corso in maniera semplice e, dal punto di vista grafico, il più fedele possibile a quella dell'applicazione Lezi.NET;
- **Apertura all'interazione vocale**: prima della fase di progettazione abbiamo effettuato un lavoro di ricerca sulla tecnologie attuali per estendere l'applicazione (sia in versione PC che in versione PDA) nel campo dell'interazione vocale.
- **Adattabilità dei contenuti**: a seconda del device usato per connettersi all'applicazione, il sistema deve essere in grado di indirizzare e rendere disponibili all'utente solo quei corsi effettivamente fruibili dal device stesso;
- **Sincronizzazione dei contenuti**: il sistema deve tenere traccia del percorso di apprendimento di ciascun utente, indipendentemente dal canale utilizzato per accedere all'applicazione, in modo da permettere il passaggio da un device ad un altro, senza dover ripetere le stesse lezioni più volte.

A conclusione della realizzazione dell'applicazione PocketLezi possiamo affermare di avere soddisfatto tutti i requisiti sopra citati. Abbiamo verificato come i dispositivi mobili abbiano ormai delle caratteristiche in grado di supportare, sebbene con le debite limitazioni, tutte le più diffuse tecnologie di progettazione Web. Siamo ora in grado di effettuare il seguente bilancio:

- **Raggiungimento di tutti gli obiettivi**: i requisiti dell'applicazione, espressi sinteticamente nel capitolo introduttivo e meglio spiegati nel capitolo 3, sono stati pienamente realizzati. L'ambiente di run-time LMS è infatti ora completamente accessibile da dispositivo mobile PocketPC. Da un punto di vista grafico siamo riusciti a ricreare un'interfaccia quasi totalmente fedele a quella

dell'applicazione Lezi.NET; questo fornisce agli utenti una completa conoscenza dell'ambiente che andranno a visitare riducendo al minimo i tempi per l'ambientazione all'interno dell'applicazione. La sincronizzazione tra i due canali di fruizione è stata pienamente realizzata da entrambi i lati di fruizione e da un punto di vista delle prestazioni non impatta significativamente con quelle dell'applicazione Lezi.NET. Le potenzialità dei PocketPC unita a quelle della tecnologia Wi-Fi potrebbero in questo senso ottenere grandi risultati nel campo dell'apprendimento a distanza.

- **Utilizzo degli standard di riferimento per l'e-learning**: dato che la piattaforma Lezi.NET realizza un learning management system compatibile con le specifiche SCORM, abbiamo cercato di mantenere tale compatibilità almeno fino a dove era possibile. Purtroppo, come visto nel capitolo 5, abbiamo dovuto abbandonare leggermente tale standard nell'ambiente di run-time per cercare un compromesso tra le limitazioni imposte dal dispositivo utilizzato e la piattaforma da utilizzare.

- **Testing ed utilizzo della piattaforma .NET**: il giudizio sull'utilizzo del framework in versione 1.1 è totalmente positivo, considerando anche la sua compatibilità con la precedente versione 1.0. 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.

- **Testing ed utilizzo del dispositivo PocketPC**: anche in questo caso le impressioni di utilizzo sono più che positive. Tale positività di giudizio riguarda in particolare l'ambiente di emulazione PocketPC, utilizzato intensivamente durante la fase di progettazione; tale emulatore ricrea fedelmente su Personal Computer un dispositivo mobile, sia come caratteristiche hardware che come caratteristiche software. In assenza di un "vero" PocketPC è stato un supporto ideale per il nostro lavoro. Tramite tale emulatore e successivamente con un "vero" PocketPC abbiamo potuto verificare come tale dispositivo abbia determinate caratteristiche di mobilità e portabilità che bene si adattano agli

ambienti di fruizione di servizi didattici. Tali caratteristiche unite alla facilità di realizzazione di applicazioni Web fruibili da dispositivi mobili non ci possono che portare ad una conclusione totalmente positiva sul progetto e sulle sue possibili evoluzioni.

Alcune estensioni possono essere apportate sia all'ambiente Lezi.NET che alla nostra applicazione PocketLezi. Innanzitutto si dovrebbe realizzare una pagina iniziale di front end che, a seconda del dispositivo utilizzo per accedere all'applicazione, sia in grado di indirizzare l'utente o verso l'applicazione PocketLezi o verso l'applicazione Lezi.NET. Una modifica potrebbe essere l'aggiunta alla piattaforma fruibile da PC di una serie di strumenti per l'inserimento e la memorizzazione delle relazioni di Mappatura tra corsi e tra SCO.
In seguito si potrebbe pensare di realizzare un sistema automatico di frammentazione di un corso, in modo tale che l'inserimento avvenga solamente in versione PC e sia poi il sistema a creare automaticamente quello in versione Pocket. Per i contenuti multimediali basterebbe creare delle pagine fatte del solo filmato e dei link al materiale; per i contenuti HTML basterebbe utilizzare un parser HTML che, analizzato il codice, sia in grado di frammentarlo in più pagine.
Per quanto riguarda l'ambiente di fruizione si potrebbe integrare un forum di discussione avanzato con la possibilità di condividere materiale didattico da parte degli studenti oltre che di scambiare opinioni, fare domande, ecc.. ed un servizio di messaggistica immediata per rendere ancora più interattiva l'esperienza di learning.

8 Bibliografia e Riferimenti

- Luciano Galliani: “La scuola in Rete”, Ed. Laterza;
- Giuseppe Mantovani: “Comunicazione e identità”, Il Mulino;
- Roberto Maragliano: “Pedagogia dell’e-learning”, Ed. Laterza;
  “Nuovo manuale di didattica multimediale”, Ed. Laterza;
- Pier Cesare Rivoltella: “EDUCATION: Modelli, esperienze, profilo disciplinare”, Carocci.

Tutti i riferimenti qui di seguito riportati sono stati verificati a Giugno 2004.

- [ADL-SCORM] Advanced Distributed Learning: http://www.adlnet.org

  Academic ADL Co-Lab: http://www.academiccolab.org/learn/
- [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/.
- [ASP.NET] The Official Microsoft ASP.NET Site: http://www.asp.net

  User Group Italiano .NET: http://www.ugidotnet.org

  Extreme .NET Forums: http://www.dotnetforums.net

  ESC Germany HomePage: http://www.escde.net

  Newsgroup: microsoft.public.it.dotnet.asp (nttp server: msnews.microsoft.com)
- [C#] C# International Standard: http://www.ecma-international.org

  Il portale italiano della programmazione C# e .NET: http://www.visualcsharp.it

  Newsgroup: microsoft.public.it.dotnet.csharp (nttp server: msnews.microsoft.com)
- [E-Learning] Specialisti in e-learning: http://www.wbt.it

  ELearningeuropa.info: http://www.elearningeuropa.info

  ELearningTouch.it: http://www.elearningtouch.it/index.php

  Enterprise Learning System: http://www.epsilonlearning.com

Mobile Learning: http://www.elearning.it/mobile/index.html

- [FRAMEWORK] Microsoft .NET Framework:
  http://msdn.microsoft.com/netframework
- [IBM PP] Ralph B. Case, Stéphane H. Maes and T. V. Raman, Position paper for the W3C/WAP Workshop on the Web Device Independent Authoring, Ottobre 2000 http://www.w3.org/2000/10/DIAWorkshop/case.htm
- [IEEE] IEEE Information Technology: http://www.ieee.org
- [IMS] IMS Global Learning Consortium Inc.: http://www.imsglobal.org

  http://www.imsproject.org
- [LMS] Learning Systems Architecture Laboratory

  http://www.lsal.cmu.edu/lsal/expertise/papers/presentations/pficordra20040219/pficordra20040219.ppt
- [MAIS] MAIS Multichannel Adaptive Information Systems:

  http://black.elet.polimi.it/mais
- [MOBILE] Mobility & Embedded Home: http://msdn.microsoft.com/mobility
- [POCKETPC] Pocket PC Developer Network: http://pocketpcdn.com/index.html
  An information hub for Pocket PC development: http://www.devbuzz.com
  http://www.microsoft.com/windowsmobile/assets/Designing_and_Building_Apps_Pocket_PC.doc
  http://www.microsoft.com/windowsmobile/assets/Pocket_PC_2002_Web_Application_Development.doc
  PocketPC Italia.com: http://www.pocketpcitalia.com
- [SALT] Speech Application Language Tags: http://www.saltforum.org
  http://www.saltforum.org/Saltforum/downloads/SALTTechnicalWhitePaper.pdf
- [SASDK] Microsoft Speech: http://www.microsoft.com/speech/
  http://www.microsoft.com/speech/docs/8928-White-Paper-final.htm
  http://216.162.203.249/downloads/BDMwpfinal.doc
  Newsgroup: microsoft.public.netspeechsdk (nttp server: msnews.microsoft.com)
- [VIRTCAMP] The Virtual Campus Project: http://www.elet.polimi.it/res/vcampus/
- [VoiceXML] VoiceXML Forum: http://www.voicexml.org

http://www.voicexml.org/specs/VoiceXML-100.pdf

- [W3C.SOAP] Simple Object Access Protocol:
  http://www.w3.org/TR/2003/PRsoap12- part1-20030507/,
  http://www.w3.org/TR/2003/PR-soap12-part2-20030507/
- [W3C.UDDI] Universal Description, Discovery and Integration:
  http://www.oasisopen.org/committees/uddi-spec/doc/tcspecs.htm
- [W3C.VoiceXML] Voice Extensible Markup Language:
  http://www.w3.org/TR/2004/REC-voicexml20-20040316/
- [W3C.WSDL] Web Service Description Language:
  http://www.w3.org/TR/wsdl/,
  http://www.w3.org/TR/wsdl12-bindings/

Ringraziamenti

Ecco finalmente il capitolo più interessante di tutto il lavoro fatto!
Per prima voglio ringraziare tutta la mia famiglia, mio papà, mia mamma, mia sorella ed i miei nonni nessuno escluso. A parte la riconoscenza per l'investimento fatto su di me, li voglio ringraziare per la piena ed incondizionata fiducia accordatami, nonostante la mia profonda reticenza nel metterli al corrente della mia carriera scolastica . Un grazie a mia sorella che ha tentato di correggere le castronerie grammaticali della tesi e che è sempre stata un ottima ed irraggiungibile lepre (anche se ancora qualche mesetto e ti riprendevo…)
Ringrazio poi gli amici di lunga data Villino (il tecnico di fiducia) e Ciccio (l'uomo con mille e più impegni), unici e stoici compagni degli innumerevoli viaggi sulla tratta Milano-Domodossola; amici sempre presenti e, cosa non meno importante, forniti di fibra ottica (e come potete immaginare per un informatico questo fatto ha sempre il suo peso). E come dimenticare K e Loca i due coinquilini creatori dei miei mille e più soprannomi che oramai mi perseguiteranno per tutta la vita.
Un grazie va a tutti i compagni di università con i quali le ore di pausa, e anche di lezione, sono trascorse in allegria e spensieratezza; un grazie speciale a Luigi compagno di lavoro di tesi e di tutti gli altri progetti dei 5 anni di studio; poi in ordine sparso ringrazio Giacomo (mio iniziatore a Scudetto), Luca (l'uomo Nintendo), Arianna (oramai dispersa tra nerboruti vichinghi) ed il gruppo dei dispersi dopo la diaspora del secondo anno ovvero i Biomedici ed i Telecom.
Un grazie speciale va a Stefano Bruna, scoperto da poco mio concittadino, il cui lungo lavoro ci ha risparmiato enooooormi fatiche.
Per ultimi, certamente non per importanza, ringrazio la prof.essa Sbattella per il suo sostegno ed il prof. Mainetti per la sua piena disponibilità e costante presenza nonostante i mille e più impegni da rispettare.

GRAZIE A TUTTI

Alessandro

I più sentiti ringraziamenti vanno alla mia famiglia, che più ha contribuito alla mia crescita e formazione: mio papà che con la sua calma e la sua bontà rappresenta un riferimento costante, mia mamma che è un supporto fondamentale per me e per tutta la famiglia e ha sempre cercato di aiutarmi in qualsiasi modo, e mio fratello che sempre più dimostra di essere anche un amico.

Non posso non ringraziare i miei compagni e amici che più hanno contribuito al raggiungimento di questo traguardo: Giacomo per le sue battute, Luca che ha sempre cercato di coinvolgermi nelle sue attività, e Laura per la sua sincera amicizia, per i momenti di divertimento che abbiamo passato insieme e per il suo aiuto morale che mi ha permesso di avere più sicurezza nei miei mezzi.

Un ringraziamento particolare va ad Alessandro, che oltre al suo fondamentale ruolo della realizzazione della tesi è sempre stato in grado di spronarmi nei momenti difficili e di riportarmi alla realtà quando era necessario come solo un buon amico sa fare.

E come dimenticare Stefano Bruna per il suo immenso lavoro su Lezi.NET e per l'aiuto fornito soprattutto agli inizi del lavoro.

Un grazie anche alla prof.essa Sbattella per i concetti base e il prof. Mainetti che si è sempre dimostrato gentile e disponibile a fornirci le linee guida per la tesi.

GRAZIE

Luigi

Indice delle Immagini

Appendice A: Manuale Utente

Il seguente manuale utente è una versione sintetica del manuale completo in primo luogo per evitare che le sezioni seguenti risultino essere eccessivamente voluminose ed 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 permettere al lettore del presente documento di orientarsi nell'applicazione in modo da poterne apprezzare appieno le funzionalità.

1. **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.

   Dopo aver inserito le proprie credenziali premere il pulsante “Login” per entrare.

2. **Default**: è la pagina visualizzata una volta effettuato il riconoscimento ed essere entrati all’interno dell’applicazione.

Nel menu visibile nella pagina è possibile avere una lista, filtrata ed ordinata in base ai criteri selezionati, dei corsi fruibili da PocketPC. Tali criteri di ordine e di filtraggio possono essere modificati selezionando il link sottostante all'elenco dei corsi. Tale collegamento visualizzerà un pannello tramite il quale si potranno visualizzare i corsi in base a:

- **Published courses**: corsi pubblicati;
- **Published courses on server**: corsi pubblicati solo presso il server sul quale ci si è autenticati;
- **Not subscribed courses**: corsi presso i quali non si hanno diritti di frequenza;
- **Preferred courses**: corsi memorizzati nella propria lista dei preferiti;
- **Subscribed courses**: corsi per i quali si hanno diritti di frequenza.

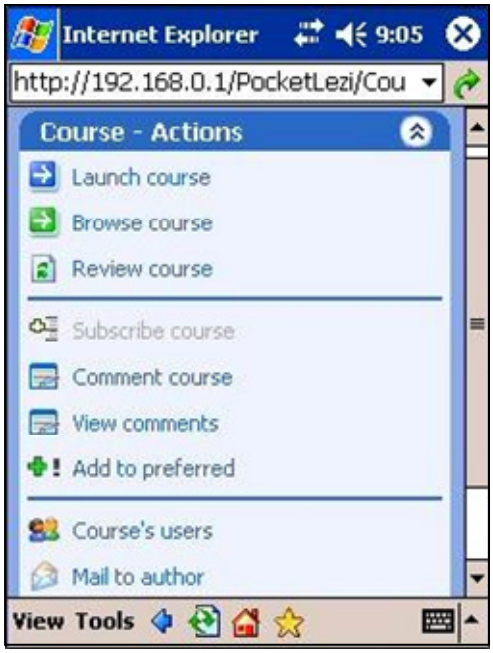
E' possibili inoltre stabilire l'ordine di visualizzazione dei corsi in base a:

- **By name**: ordinamento alfabetico crescente in base al 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.

Tramite il pulsante "Logout" è possibile uscire dall'applicazione.

Cliccando sul nome di un corso è possibile visualizzarlo.

3. **Corso**: la pagina del corso visualizza un pannello contenente una serie di operazioni fattibili dall'utente inerenti al corso.



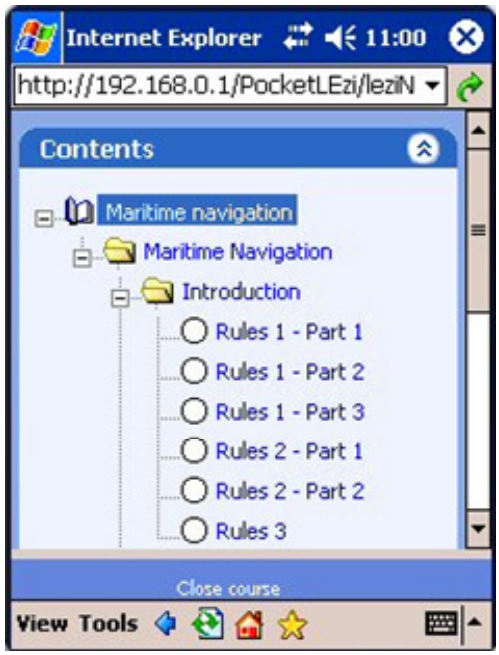
Le voci di tale lista sono disabilitate quando il link è di colore grigio, sono abilitate quando il link è attivo e di colore azzurro. L'abilitazione di una voce è dovuta al ruolo ricoperto dall'utente, allo stato attuale del corso ed ai diritti che l'utente ha sul corso. Le azioni eseguibili sono:

- **Launch course**: Esegue un corso. Se il corso è SCORM compatibile, 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 che il sistema tenga traccia delle azione dell'utente.
- **Review course**: Permette di vedere un corso e lo stato raggiunto all'ultimo Launch senza 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 sul corso. Per poter fare la richiesta di sottoscrizione è necessario compilare una form che richiede la motivazione della richiesta.

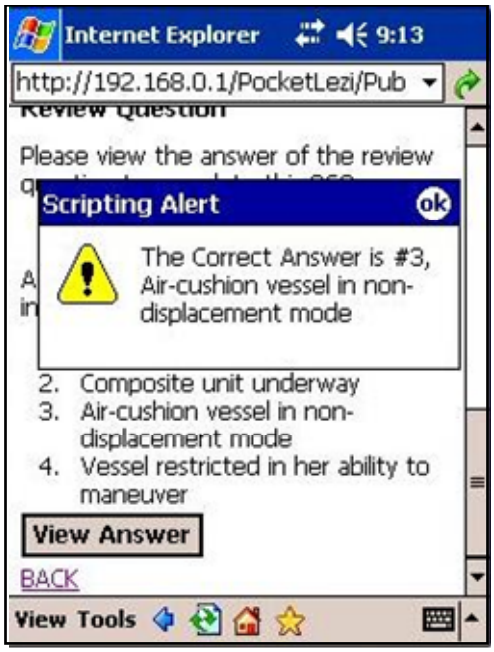
- **View comments**: Mediante questa voce è possibile ottenere una lista di commenti che gli altri utenti hanno fatto sul corso.
- **Add to preferred**: Link per poter aggiungere il corso corrente alla propria lista di corsi preferiti.
- **Course's users**: Visualizza un elenco di utenti che hanno diritti 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'esecuzione del viewer dei contenuti.



Premendo il link "Close course" è possibile uscire dall'ambiente di fruizione e tornare alla pagina di default. Selezionando invece uno dei contenuti, verrà

visualizzata la pagina desiderata. Da tale pagina si potrà decidere se utilizzare il link "Back" per tornare all'albero dei contenuti interrompendo la lezione, oppure di visualizzare la risposta alla domanda posta decidendo di concludere la lezione.



A seconda del link seguito per tornare all'albero dei contenuti, quest'ultimo visualizzerà lo stato della lezione appena seguita.