UNIVERSITÀ DEGLI STUDI DI MILANO
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea triennale in Informatica

ACCESSO MULTIMODALE A SERVIZI DI E-LEARNING

Relatore: Prof.ssa Silvana CASTANO

I° Correlatore: Ing. Thimoty BARBIERI

II° Correlatore: Dott. Massimo NAZZARI

Tesi di Laurea di:

Alessio SOMASCHINI

Matr. Nr. 639060

Anno accademico 2003/2004

INDICE

INTRODUZIONE

L'argomento della tesi nasce dall'idea di mettere a disposizione un servizio di e -learning con accesso multimodale. L'utente deve avere la possibilità di accedere sia in modalità vocale che manuale (mouse,tastiera). In particolare l'obiettivo è di creare un servizio per utenti ipovedenti, ovvero utenti con notevoli difficoltà visive ma in grado di interagire manualmente.
Le attività relative all'argomento sono state svolte presso il Laboratorio di Ricerca e Sviluppo Multimediale del Politecnico di Milano, nell'ambito di uno stage. Infatti come base di partenza per l'obiettivo prefisso, è stata selezionata una piattaforma web già esistente: Lezi.NET. Si tratta sostanzialmente di un sistema per la pubblicazione e fruizione di contenuti didattici in rete sviluppata da uno studente del Politecnico.
L'applicazione MultiLezi.NET ha l'obiettivo di mettere a disposizione un canale separato per l'accesso multimodale alla fruizione dei contenuti didattici della piattaforma Lezi.NET. Pur trattandosi di un canale separato il requisito fondamentale è quello di mantenere sincronizzati gli accessi alla piattaforma, per tenere traccia dei percorsi formativi dell'utente indipendentemente dal canale utilizzato per la fruizione dei contenuti.

La realizzazione di MultiLezi.NET si colloca all'interno del progetto MAIS (Multichannel Adaptive Information System) [11], che ha come obiettivo lo sviluppo di modelli, metodi ed applicazioni che permettano l'implementazione di Sistemi Informativi Adattativi Multicanale.
Il lavoro rientra nella sperimentazione di servizi adattativi secondo il modello di riferimento MAIS, con particolare attenzione al supporto a disabilità sensoriali (studenti ipovedenti).
L'aspetto fondamentale è quindi la multimodalità, in altre parole la possibilità di attuare diverse modalità d'interazione che faciliterà l'accesso a studenti ipove denti.

Le diverse fasi del lavoro svolto sono descritte nel presente documento e organizzate come segue :

E-learning e standard per l'e -learning :
La prima fase del lavoro ha richiesto lo studio del dominio dell'applicazione, l'e -learning, per avere una visione chiara dei sistemi e degli standard disponibili. In particolare sono stati approfonditi gli

argomenti riguardanti il sistema per l'e-learning LMS (Learning Management System), e gli standard IMS e SCORM utilizzati per la realizzazione di Lezi.NET.
Multimodalità e tecnologie per l'accesso vocale :
La multiodalità rappresenta un altro argomento chiave del lavoro, e a tal proposito è stata effettuata un'attività di ricerca soprattutto per capire i principi che guidano lo sviluppo di applicazioni multimodali e la relazione tra multimodalità e disabilità visive.
E' stato inoltre necessario effettuare una ricerca tecnologica nell'ambito dell'interfacciamento vocale, che ha portato a prendere in considerazione principalmente SALT e VoiceXML.

Il contesto applicativo e obiettivi dell'applicazione :
Il punto di partenza per lo sviluppo di MultiLezi.NET è costituito dalla piattaforma di e-learning Lezi.NET; è stato quindi necessario lo studio delle funzionalità e dell'architettura di questa piattaforma.
L'obiettivo principale è di realizzare un servizio per utenti ipovedenti in un ambiente di e-learning. Un altro obiettivo è quello di realizzare un canale separato (MultiLezi.NET) per l'accesso alla piattaforma Lezi.NET, garantendo la sincronizzazione degli accessi in modo da tener traccia dei percorsi formativi degli utenti indipendentemente dal canale utilizzato.
L'analisi delle tecnologie per l'interfacciamento vocale ha portato a mettere a confronto VoiceXML e SALT, per individuare quella più adatta allo scenario delineato dagli obiettivi prefissi.

Analisi e progettazione :
La fase di analisi ha richiesto la definizione dei requisiti generali per la realizzazione dell'interfaccia multimodale. Successivamente è stata fatta l'analisi dei requisiti funzionali dell'applicazione MultiLezi.NET : accesso alla piattaforma, consultazione dei corsi e fruizione dei corsi.
L'attività di progettazione ha portato a definire la modalità d'integrazione con Lezi.NET, gli standard da utilizzare per l'e-learning, le tecnologie da utilizzare per lo sviluppo e la struttura dell'applicazione web.

Implementazione :
La descrizione dell'attività d'implementazione spiega le soluzioni adottate per la realizzazione di MultiLezi.NET.
E' stata quindi descritta la struttura delle principali pagine dell'applicazione che possono essere raggruppate nelle seguenti aree : autenticazione, homepage dell'applicazione, visualizzatore dei corsi, configurazione.
Assumono particolare importanza le soluzioni adottate per implementare i vari aspetti dell'interfaccia vocale : navigazione della pagina guidata da output vocali, comandi vocali per le principali funzioni e fruizione basata sulla lettura del contenuto da parte dell'applicazione.

Esempi d'uso e valutazione dei risultati :

Gli esempi d'uso permettono di dimostrare il corretto funzionamento di MultiLezi.NET, attraverso la presentazione delle schermate principali. Per ogni area applicativa sono state scelte delle schermate rappresentative delle principali funzioni, anche se una dimostrazione completa dell'applicazione avrebbe richiesto un supporto audio e non solamente cartaceo.

Al fine di valutare la reale efficacia dell'applicazione, sono stati infine effettuati dei test coinvolgendo utenti ipovedenti.

1 E-LEARNING E STANDARD PER L'E-LEARNING

L' e-learning è una metodologia didattica che offre la possibilità di erogare contenuti formativi elettronicamente, attraverso Internet o reti Intranet (ASFOR , 2003). Per l'utente rappresenta una soluzione di apprendimento flessibile, personalizzabile e facilmente accessibile.

La Formazione a Distanza viene definita come "l'insieme dei metodi didattici in cui, a causa della separazione fisica tra gli insegnanti e i discenti, la fase interattiva dell'insegnamento (stimolo, spiegazione, domande, guida), come inoltre quella pre-attiva (scelta degli obiettivi, compilazione del curriculum e delle strategie didattiche), è condotta per mezzo cartaceo, meccanico, elettronico" (M. Moore in G. Costa, E. Rullani, 1999).

ISFOL pone una definizione di Fad (Formazione a distanza) più sintetica, indicandola come "tutte le varie forme di autoistruzione interattiva che si realizzano attraverso canali di comunicazione multimediali" (ISFOL, 1994).

La Fad viene suddivisa in tre livelli di evoluzione corrispondenti allo sviluppo degli strumenti di supporto della sua divulgazione (G. Trentin, 2001):

- Formazione per corrispondenza : tramite servizi postali;
- Formazione plurimediale : caratterizzati da un uso integrato del materiale a stampa, trasmissioni televisive e registrazioni sonore;
- Formazione in rete : tramite reti internet, extranet ed intranet.

La forma più evoluta di formazione a distanza (formazione in rete), anche definita on-line learning, permette l'interazione dei partecipanti al fine di superare l'isolamento del singolo.

L'on-line learning costituisce solo una parte dell'insieme delle attività formative sottostanti il concetto di e-learning. Infatti il termine e-learning copre una gamma di applicazioni e di processi che si avvalgono di mezzi di tipo elettronico e include :

- la rete internet;
- le reti intranet ed extranet;
- la trasmissione via satellite;
- l'utilizzo di materiale audiovisivo analogico;
- la televisione interattiva;
- l'uso di cd-rom.

Le differenze principali presenti fra istruzione tradizionale e quella a distanza sono quindi i diversi canali di comunicazione: voce con contatto faccia a faccia fra studente e docente e fra studenti, contro trasmissione bidirezionale di informazioni in modo sincrono o asincrono dipendente tuttavia dalla tecnologia disponibile.

Mediante il miglioramento tecnologico si cerca di portare all'efficienza e naturalezza della divulgazione verbale i canali di comunicazione della FaD. L'avvento delle nuove tecnologie, in primo luogo di internet, ha dato la possibilità di rendere l'apprendimento a distanza maggiormente interattivo e dinamico.

L'e-learning comporta una serie consistente di vantaggi sia per lo studente che per il docente :

- Facilità di raggiungimento delle risorse : qualsiasi tipo di contenuto che sia statico od interattivo può essere memorizzato in formato elettronico e quindi essere distribuito facilmente in rete;
- Risparmio di costi e di tempo : da parte di chi produce i contenuti viene a mancare la necessità di trovare luoghi adatti per l'insegnamento quali aule e sale e da parte dello studente non è più necessario recarsi fisicamente presso tali luoghi con grande risparmio di tempo e denaro;
- Personalizzazione : lo studente può scegliere il tipo di contenuto più adatto al suo meccanismo di apprendimento, fra i diversi messi a disposizione;
- Facilità d'aggiornamento : possibilità di cancellare, modificare o inserire lezioni in qualsiasi momento, in modo da calibrare costantemente i corsi sulle reali necessità degli utenti.
- Misurazione livello apprendimento : è possibile seguire l'andamento dell'apprendimento dello studente, in modo continuo ed economico, magari con delle verifiche in determinati punti del percorso formativo;
- Vantaggi sociali : tutti devono avere la possibilità di apprendere, compresi gli individui impossibilitati a muoversi, geograficamente isolati o socialmente svantaggiati;

Si possono però individuare anche degli svantaggi nel modello dell'e-learning :

- Limiti ipotetici legati alla banda di connessione : limiti della banda a disposizione per i collegamenti significano deboli e lente prestazioni per quanto riguarda il suono, il video, le animazioni e la grafica "pesante". Ovviamente i maggiori problemi si hanno sulla rete Internet pubblica, piuttosto che sulle Intranet aziendali. La diffusione crescente della banda larga è comunque di conforto per la risoluzione di questo problema;
- Mancanza di contatto umano : la comunicazione attraverso i mezzi tecnologici fa venire meno il contatto diretto tra individui; questo si traduce in una scarsa qualità di comunicazione tra

docente e studente rispetto a quella ottenibile in un contatto diretto, e in mancanza di rapporto sociale tra studenti soprattutto in termini di esperienze vissute in classe.

Uno dei sistemi tecnologici a servizio dell'e-learnig è LMS (Learning Management System), e proprio questo sistema è stato adottato per la realizzazione di Lezi.NET. I paragrafi seguenti presentano la descrizione dei principali standard per l'e-learning.

1.1 Learning Management System (LMS)

Il termine Learning Management System , di solito indicato con la sigla LMS, indica un insieme di funzionalità progettate sia per distribuire e gestire contenuti che per interazioni e progressi degli utenti.

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

Un modello generale che mostra i componenti e i servizi potenzialmente presenti in un implementazione di un LMS è visibile in Figura 1.

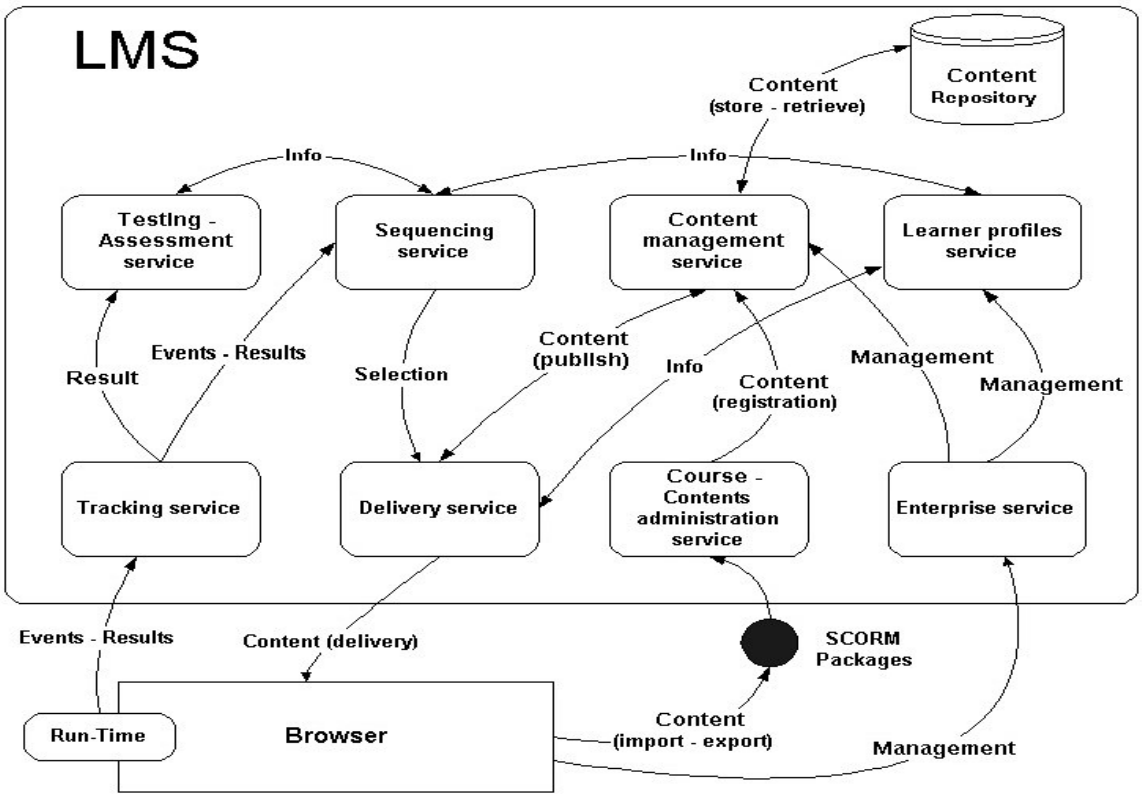

Figura 1: Componenti e servizi di un LMS generico

Segue una breve descrizione dei componenti dell'LMS generico di Figura 1.

- Run-Time **:** è l'ambiente in cui il contenuto viene eseguito. Permette l'interazione fra componente di presentazione, generalmente un browser, e LMS quando la comunicazione è iniziativa del contenuto. I contenuti eseguibili in un Run-Time possono comunicare attivamente, tramite logica di controllo contenuta negli stessi, con il sistema. L'ambiente Run-Time fornirà quindi delle API che potranno essere usate per interagire con l'LMS. Ad esempio, una volta che il contenuto in esecuzione è terminato, tramite l'API del Run-Time è possibile comunicare tale evento di stato in modo che l'LMS sia in grado di determinare l'azione successiva come ad esempio la messa in esecuzione del contenuto successivo. Le API forniscono, generalmente, anche la possibilità di ottenere e/o salvare informazioni relative allo studente e allo stato della fruizione corrente come ad esempio tempi d'esecuzione, risultati di tests etc. Esistono diverse soluzioni implementative per realizzare tale componente: si può utilizzare una applet client-side eseguita dal browser che si occupa di tradurre le richieste dei contenuti verso l'LMS oppure un componente JScript che, tramite Web Services, giri le chiamate del contenuto all'LMS.
- Delivery service : servizio che mette a disposizione i contenuti all'ambiente di esecuzione, tramite chiamate al Content management service. Per sapere quale contenuto visualizzare viene interrogato il Sequencing service.
- Tracking service : servizio che si occupa di mettere in comunicazione l'ambiente di esecuzione con il resto del sistema. Fornisce l'implementazione server-side delle API che il contenuto può utilizzare nel Run-Time. Una delle funzioni principali di questo modulo è quella di tracciare lo stato del contenuto (inizializzato, completato, terminato, errore, ecc.) e di registrare informazioni sulla fruizione (risultato di un test, tempi di esecuzione, ecc.).
- Sequencing service : servizio che, in base allo stato del contenuto visualizzato (terminato correttamente, terminato non correttamente, ecc.) e allo stato dell'utente (possibilità o meno di fruire un contenuto) decide quale sarà il prossimo contenuto da presentare all'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 utenti.

- Learner profile service : servizio che gestisce tutte le informazioni dell'utente non relative ad un contesto di esecuzione; ad esempio preferenze riguardanti la lingua dell'utente.
- Enterprise service : servizio per la gestione di utenti, gruppi, access control lists, permessi, autorizzazioni ed autenticazione.
- Content repository : repository dei contenuti. Per effettuare operazioni di inserimento di contenuti, occorre definire un formato comune a vari tipi di LMS, per garantire interscambio di contenuti ed aggregazione di contenuti.

Una caratteristica dell'architettura presentata in Figura 1, è l'alta modularità che facilita la realizzazione del sistema in ambiente distribuito.

Inoltre i contenuti dovrebbero essere organizzati in unità di piccole dimensioni, aggregando le quali è possibile ottenere capitoli, moduli, corsi. Per favorire il riutilizzo delle singole untà, queste dovrebbero essere le più indipendenti possibili dal contesto in cui vengono utilizzate. Una regola da seguire per ottenere questa indipendenza impone che una unità non possa lanciare un'altra unità di contenuto. Tale operazione è compito dell'LMS.

1.2 IMS

IMS Global Learning Consortium [8] sviluppa e promuove l'adozione di specifiche tecniche per la realizzazione di sistemi di learning caratterizzati dall'interoperabilità.

Le specifiche prodotte possono essere suddivise come segue :

- Specifiche per la descrizione, ricerca e scambio di contenuti :
    - Meta-Data : specifiche per l'associazione di elementi descrittivi ai contenuti, col fine di facilitarne la ricerca e la distribuzione;
    - Content packaging : descrivono le strutture dati con le quali i contenuti dovrebbero essere organizzati per garantire interoperabilità tra diversi LMS; lo scopo è quello di fornire un set standardizzo di strutture per lo scambio di contenuti, in modo che un contenuto creato da un sistema possa essere mandato in esecuzione su un altro sistema;
    - Question and Test : riguarda la definizione di strutture per la descrizione di domande e tests; inoltre mette a disposizione un meccanismo per il reporting dei risultati;
    - Digital Repositories Interoperability : definisce un insieme di funzione che ogni repository dovrebbe implementare per favorire l'interoperabilità con altri sistemi.
- Specifiche per l'interazione ed il tracking dei contenuti :

  - Simple Sequencing : definisce un metodo per stabilire la sequenza dei contenuti, tenendo presente l'interazione dell'utente con il contenuto in esecuzione;
  - Competencies : definisce un modello per le competenze associate a contenuti ed utenti.
  - Learning Design : definisce un framework ed un linguaggio per la descrizione di esperienze di learning, da condividere con altri sistemi;
  - Accessibilità : definisce linee guida per lo sviluppo di interfacce utente e contenuti accessibili agli utenti utilizzando diversi sistemi di accesso;
- Specifiche per l'interoperabilità applicativa :
  - Learner Information Package : definisce una struttura per l'organizzazione di informazioni relative agli utenti da condividere tra sistemi di learning e sistemi amministrativi;
  - Enterprise : definisce le modalità per il trasferimento di informazioni organizzative relative a studenti e gruppi di studio, tra diversi sistemi applicativi.

Dall'insieme di specifiche messe a disposizione dall'IMS, Lezi.NET implementa solo le specifiche relative al Content Packaging. Per altri sottoinsiemi sono state utilizzate le specifiche SCORM, come sarà spiegato in seguito. Il paragrafo seguente descrive con maggior dettaglio le specifiche IMS Content Packaging.

1.2.1 Content Packaging specification

Lo scopo delle specifiche Content Packaging è focalizzato sulla definizione dell'interoperabilità tra sistemi che necessitano di effettuare import, export, aggregazione e disaggregazione di packages di contenuti. La necessità di disporre di un set di strutture standard da utilizzare per creare e scambiare packages di contenuti ha portato alla definizione del modello concettuale mostrato in Figura 2.

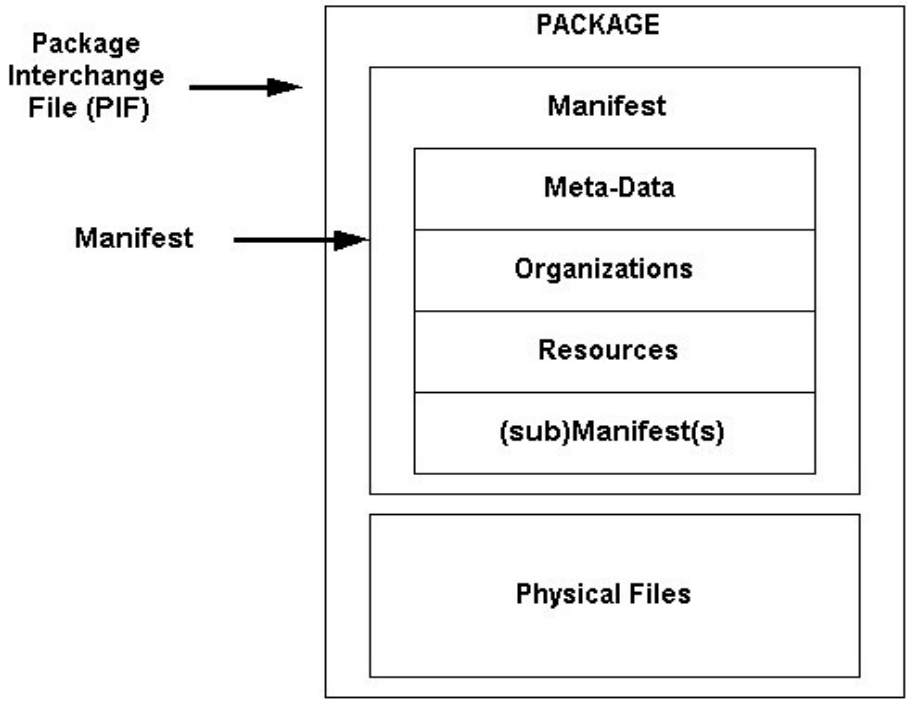

Figura 2 : IMS Package

Il Package è composto da due parti fondamentali : un file XML speciale che descrive l'organizzazione del contenuto e delle risorse del package; i file fisici dei contenuti descritti dall'XML. Il file XML è chiamato IMS Manifest file, mentre il file che contiene l'intero package è chiamato Package Interchange File.

Segue una descrizione delle relazioni tra le parti presenti in Figura 2 :

- Package Interchange File : un singolo file (.zip, .rar, .jar) che contiene un top-level manifest file chiamato imsmanifest.xml, e tutti i file fisici identificati dal manifest;
- Package : una directory logica contenente un file XML, tutti gli eventuali file per la gestione dell'XML (DTD or XSD file) ed i file delle risorse eventualmente or ganizzati in sottodirectory;
    - Top-level Manifest : un file XML che descrive il contenuto del package; può contenere altri (sub)Manifests; ogni istanza di manifest è così composta :
        - Meta-Data section : elemento XML che descrive l'intero manifest;
        - Organizations section : elemento XML che descrive zero, una o multiple organizzazioni del contenuto del manifest;

- ♣ Resources section : elemento XML contenente riferimenti alle risorse del manifest, nella forma di meta-dati per la descrizione delle risorse e riferimenti a file fisici esterni;
- ♣ (sub)Manifest : zero, uno o più manifest nidificati;

- Physical Files : sono i vari file fisici (di testo, grafici, multimediali) descritti dai manifest(s)

Un Package rappresenta un'unità di contenuto riutilizzabile; potrebbe essere una parte di un corso, un corso o addirittura una collezione di corsi. Un Package non deve essere necessariamente inserito in un Portable Interchange File, ma può essere distribuito su supporti come CD-ROM. Il requisito da rispettare è che l'imsmanifest .xml ed i relativi file di supporto (DTD,XSD) devono essere nella root del supporto di distribuzione.

1.2.2 IMS Manifest

Questo paragrafo da una visione concettuale e descrittiva degli elementi contenuti in un manifest. La figura seguente illustra gli elementi principali di un manifest e le relazioni che possono intercorrere tra essi.

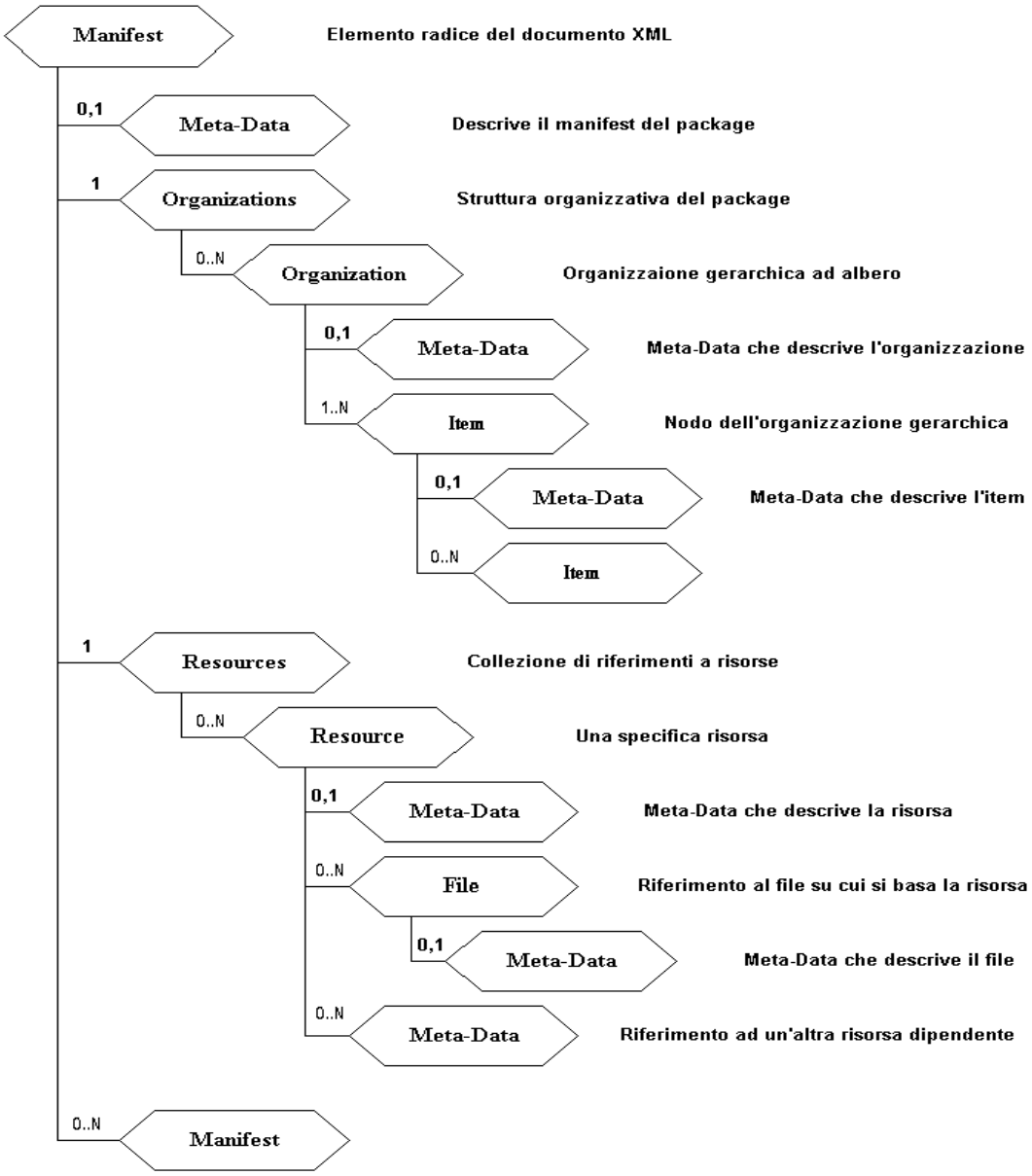

Figura 3: Struttura IMS Manifest

Come si può vedere in figura, accanto ad ogni elemento c'è una notazione che sta ad identificare la relazione che intercorre tra gli elementi collegati :

- 1 : uno a uno;
- 0,1 : zero o uno a uno;
- 0..N : zero o uno a molti;
- 1..N : uno a molti;

Elementi come Organization, Item e Resource, possiedono l'attributo identifier che indica l'identificatore univoco dell'elemento. Tale univocità è valida solo all'interno del manifest setsso. L'element Item possiede inoltre l'attributo identieferref che permette di puntare direttamente alla risorsa che rappresenta l'item.

Un elemento Organization può essere rappresentato in XML come segue :

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

Un LMS interpreterebbe il codice sopra, come segue :

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

Un elemento Resources può invece essere rappresentato come segue :

```
<resources>
  <resource identifier="RESOURCE1" type="webcontent" href="sco01.html">
```

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

Per una descrizione più dettagliata degli elementi dell'IMS Manifest si rimanda al riferimento bibliografico [8].

1.3 ADL SCORM

L'iniziativa Advanced Distribuited Learning (ADL) nasce nel 1997 da una collaborazione tra Department of Defense (DoD) e White House Office of Science e Technology Policy (OSTP). Lo scopo di tale iniziativa è di stabilire un ambiente di leraning distribuito, che permetta l'interoperabilità di tool di learning e contenuti su larga scala.

Al fine di raggiungere tali obiettivi è stato definito un modello di riferimento SCORM (Sharable Content Object Reference Model) [1], un insieme di standard, specifiche e linee guida per contenuti e sistemi di learning. Tali specifiche possono essere suddivise in due gruppi :

- Content Aggregation Model : contiene indicazioni per identificare ed aggregare le risorse in uno strutturato contenuto di apprendimento, ed è suddiviso ulteriormente in :
    - Content Model : nomenclatura per la descrizione dei vari componenti dei contenuti di apprendimento;
    - Content Packaging : definisce come aggregare le risorse di learning per l'interscambio tra diversi ambienti;
    - Meta-Data : meccanismo per la descrizione di specifiche istanze dei componenti del Content Model;
    - Sequencing and Navigation : definisce un insieme di regole per la sequenza e l'ordinamento delle attività.
- Run-Time Environment : contiene le specifiche per la definizione di un ambiente che permetta il lancio dei contenuti, la comunicazione con un LMS ed il tracking di eventi e dati (risultati di tests, tempi di esecuzione, ecc.); si possono individuare le due sottosezioni :
    - Launch : un modo comune., per i vari LMS, di lanciare i contenuti;
    - Run-Time API : meccanismo di comunicazione per informare l'LMS sullo stato dell'esecuzione; anche utilizzato per reperire o memorizzare dati tra il contenuto in esecuzione e l'LMS;
    - Data Model : insieme di elementi utilizzati per definire le informazioni relative al contenuto che devono essere tracciate (stato dell'esecuzione, risultato di un test, ecc.).

1.3.1 Content Aggregation Model

Le specifiche contenute in questo documento, permettono di creare dei contenuti composti da diverse risorse per l'apprendimento. Il Content Model definisce la nomenclatura dei vari componenti come segue :

- Assets : sono l'elemento elementare nel dominio dell'e -learning; possono essere visti come la rappresentazione elettronica di testi, suoni, video, pagine web ecc. ;
- Sharable Content Object (SCO) : è una collezione di uno o più assets che generalmente include uno specifico asset (ad esempio una pagina html con codice Jscript) che permette l'utilizzo del Run-Time environment per la comunicazione con l'LMS. Lo SCO rappresenta il più basso livello di granularità che una risorsa di learning, interagente tramite le API con l'LMS, possa avere. Per garantire la riusabilità, uno SCO dovrebbe essere indipendente dal contesto in cui

viene creato, in modo che possa essere riutilizzabile in altri contesti insieme ad altri SCO creati in contesti ancora diversi. L'interindipendenza fra SCO impone un ulteriore vincolo: uno SCO non deve mai fare riferimento ad un altro SCO tramite, ad esempio, un hyperlink. Questo perché se uno SCO potesse lanciare un altro SCO sarebbe scavalcato il meccanismo di launch del LMS, compromettendo il corretto funzionamento del Run-Time environment. SCORM non impone limiti alle dimensioni degli SCO, tuttavia consiglia di suddividere un contenuto di grandi dimensioni in unità logicamente indipendenti e di creare diversi SCO per ognuna di queste unità; aggregando gli SCO prodotti si arriva ad ottenere il contenuto iniziale.

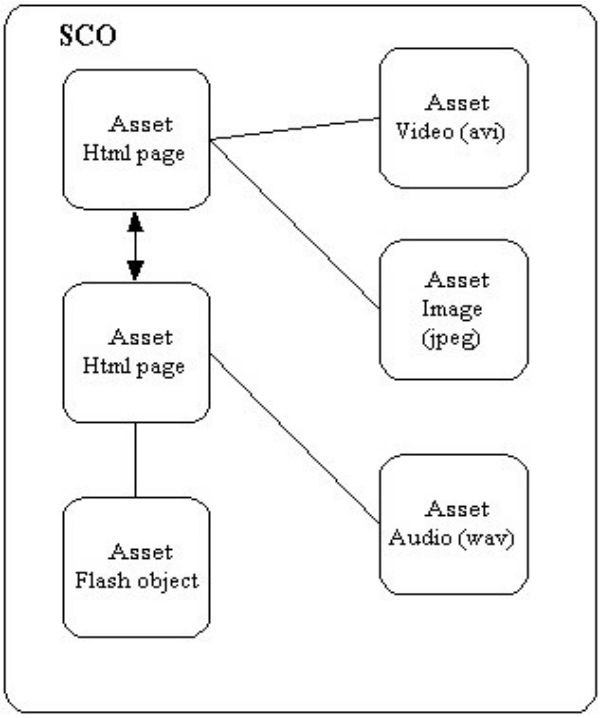

Figura 4 : Esempio di SCO contenente diversi assets; la navigazione fra i diversi assets è possibile, ed è compito dello SCO fornire gli hyperlink necessari

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

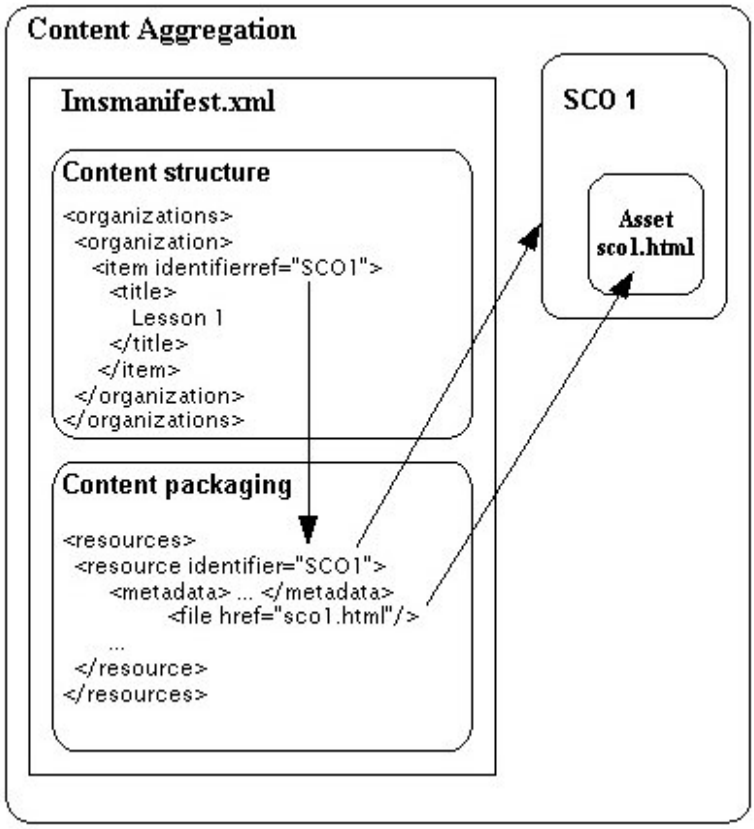

Figura 5 : Content Aggregation contenente un SCO con un asset. Il manifest descrive la struttura interna della content aggregation. La content structure descrive il modo in cui i contenuti verranno presentati, mentre la content packaging fornisce i riferimenti agli SCO e assets

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

- le Content Packaging, estensione delle specifiche IMS Content Packaging, 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 all'organizzazione delle risorse e ai riferimenti ai files degli assets.
- le Meta-Data dictionary che definiscono dei meta-dati per ognuna delle tre precedenti entità in modo da facilitare le operazioni di archiviazione, ricerca e riutilizzo delle risorse ad ogni livello di granularità. I meta-dati sono rappresentati mediante elementi presenti nel manifest e documenti xml standalone associati a ciascuno SCO. Un riferimento a tali files è in ogni modo presente nel manifest.

1.3.2 Run-Time Environment

Al fine di poter garantire il riutilizzo delle risorse e l'interoperabilità fra diversi LMS è necessario definire un unico ambiente d'esecuzione che fornisca una interfaccia comune per poter permettere l'esecuzione delle risorse e la comunicazione di queste ultime con l'LMS.
Il Run-Time environment fornisce tre tipi di entità per soddisfare tali requisiti: un meccanismo di lancio, launch, un'interfaccia di comunicazione, API, ed un modello di dati, data model. Nella Figura 6 è possibile osservare le relazioni che intercorrono fra le suddette entità.

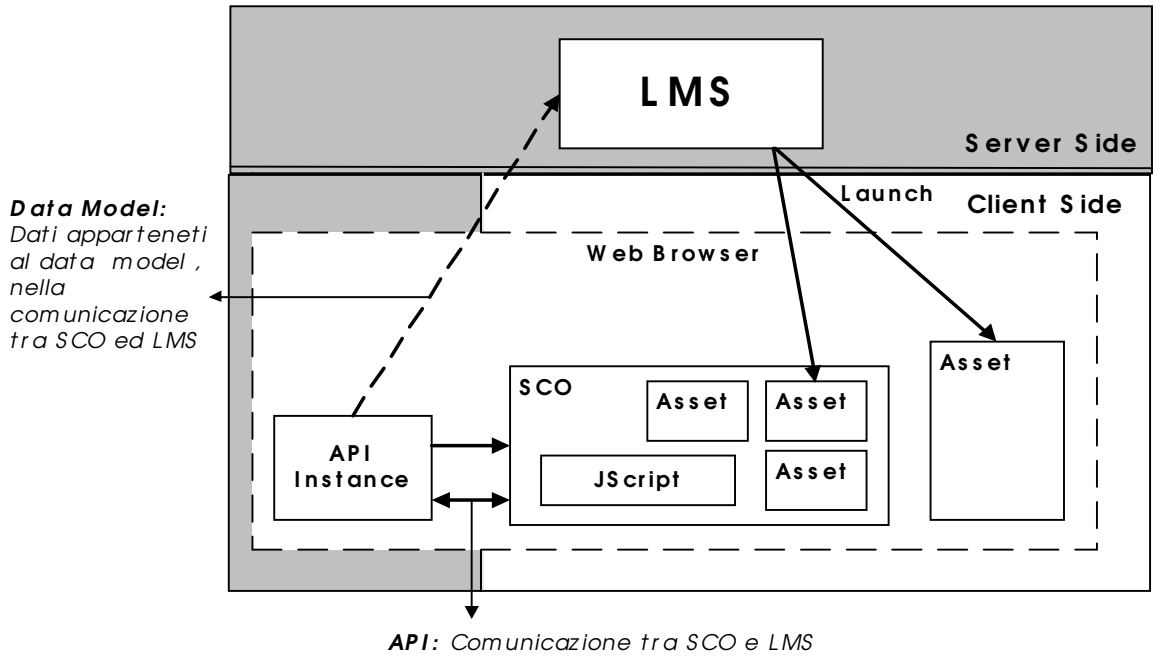

Figura 6 : Modello concettuale del Run-Time environment

Il processo di Launch definisce una modalità comune per il lancio di contenuti Web-based; i contenuti che possono essere lanciati secondo il modello SCORM, possono essere SCO o Asset. Il meccanismo di comunicazione tra SCO e LMS è standardizzato da un set di API. Il Data Model definisce gli elementi del modello di dati che l'LMS e lo SCO si aspettano di conoscere. E' compito dell'LMS tener

traccia degli elementi del data model relativi ad un determinato SCO attraverso le sessioni dell'utente, e lo SCO deve utilizzare solo tali elementi del data model per garantire il riutilizzo attraverso diversi sistemi.

1.3.3 Interfaccia di comunicazione (API)

L'utilizzo di un'interfaccia comune, oltre a garantire interoperabilità e riusabilità de lle risorse, premette di semplificare il processo d'implementazione di un sistema di learning completo. Infatti, lo sviluppatore dei contenuti non dovrà preoccuparsi dei dettagli implementativi del LMS e, viceversa, lo sviluppatore di un LMS dovrà limitarsi a fornire, nell'ambiente di esecuzione, le API.
L'API Adapter è una porzione di codice che implementa le API è che viene fornita alla risorsa dal Run-Time generalmente mediante un oggetto JScript, con identificatore API , che viene cercato nella struttura gerarchica dei frames del browser dalla risorsa stessa, dopo che è stata lanciata dal LMS. Tale modulo può essere considerato come parte client-side del LMS eseguito nel browser. Una volta che uno SCO è stato lanciato, tramite le API, potrà chiamare un insieme di otto metodi :

- Initialize(): 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;
- Terminate()**:** lo SCO esegue questa chiamata quando ritiene che il canale di comunicazione con il LMS non sia più necessario e che la propria esecuzione possa essere terminata;
- GetValue(): questo metodo permette allo SCO di ottenere informazioni dal Data Model gestito dall'LMS;
- SetValue(): questo metodo permette allo SCO di scrivere nel Data Model;
- Commit(): lo SCO chiama questo metodo quando desidera che le precedenti operazioni di scrittura (mediante metodo SetValue()) debbano essere considerate valide e salvate in modo da garantire persistenza;
- GetErrorString(): permette di ottenere una descrizione testuale di un errore espresso in formato numerico passato come parametro al metodo;
- GetLastError(): ritorna un numero indicante un eventuale condizione d'errore rilevata dall'LMS durante l'esecuzione dello SCO e quindi delle varie chiamate che quest'ultimo può aver fatto;

- GetDiagnostics(): necessario per poter supportare una descrizione degli errori dettagliata, specifica di un particolare LMS (mentre i tipi d errori ritornati da GetErrorString sono appartenenti allo standard).

Gli stati in cui si può trovare uno SCO sono sostanzialmente 3 :

- Not Initialized : è lo stato in cui si trova lo SCO prima del lancio del metodo Initialize(); è responsabilità dello SCO trovare l'API e lanciare il metodo Initialize();
- Running : è lo stato in cui si trova lo SCO dopo il lancio del metotodo Initialize() e prima del lancio del metoto Terminate(); in questo stato lo SCO può lanciare qualsiasi metodo messo a disposizione dalle API, tranne Initialize().
- Terminated : è lo stato in cui ritrova lo SCO dopo il lancio del metodo Teriminate(); lo SCO può lanciare i metodi per la gestione degli errori e diagnostici.

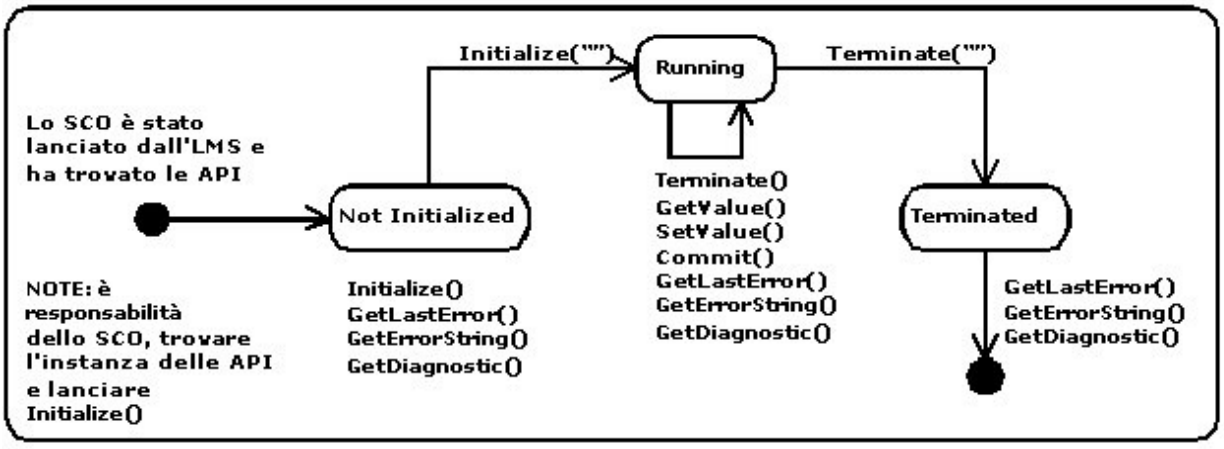

Figura 7: Diagramma degli stati di uno SCO

1.3.4 Data Model

Lo scopo di definire un modello di dati comune è di assicurare che un determinato insieme di informazioni relative agli SCO, possa essere tracciato da diversi LMS.

L'insieme dei dati rappresentati dal modello include, informazioni relative all'utente, interazioni tra utente e SCO, stati di successo e completamento.

Il modello è rappresentato da una struttura ad albero avente come radice *cmi*, dalla quale partono tutti i rami relativi alla descrizione dei vari tipi di elemento. Per indicare gli elementi del modello si utilizzano dei nomi espressi in dot-notation (es. cmi.success_status).
Ogni foglia della struttura ad albero è un tipo di dato con delle regole di accessibilità (read/write). Ad esempio la stringa *cmi.core.lesson_status* sta ad indicare un particolare tipo relativo allo stato della lezione.
I metodi SetValue() e GetValue() delle API, lanciati dagli SCO per interagire con il Data Model, utilizzano delle stringhe in dot-notation per riferirsi agli elementi interessati. Ad esempio uno SCO potrebbe leggere e scrivere il proprio stato nel modo seguente :

```
var status = API.GetValue("cmi.core.lesson_status");
API.SetValue("cmi.core.lesson_status","completed");
```

2 MULTIMODALITÀ E TECNOLOGIE PER L'ACCESSO VOCALE

2.1 Multimodalità

La comunicazione multimodale è spesso intesa come l'opportunità di offrire diversi canali di accesso all'informazione, considerando i diversi contesti in cui un utente si può cala re (in ufficio, in macchina). Da un altro punto di vista la multimodalità può essere usata non solo per accedere ad un'informazione in qualsiasi momento e ovunque, ma anche come mezzo per comunicare la stessa informazione all'utente in più di una forma (vi siva, audio, tattile) allo stesso tempo, con lo scopo di aiutarlo nel superamento di difficoltà relative a disabilità cognitive, sensoriali e fisiche.
Il documento W3C "Multimodality Interaction Activity" [20], che studia come supportare la ricezione, la produzione e la sincronizzazione di modalità ridondanti, ha introdotto una distinzione tra multimodalità supplementare e multimodalità complementare. Un'applicazione fa un uso supplementare della multimodalità se permette di portare a termine ogni interazione, utilizzando una delle modalità disponibili, come se fosse l'unica disponibile. In questo modo l'applicazione mette l'utente in condizione di poter scegliere la modalità più adatta per il tipo di interazione e per la situazione. Dall'altra parte, un'app licazione che fa uso complementare della multimodalità, potrebbe non essere accessibile separatamente con le modalità a disposizione; in questo caso all'utente non è permesso scegliere la modalità da usare per l'interazione (T. Barbieri, A. Bianchi e L. Sb attella. *Multimodal communication for vision and hearing impairments*. set. 2004).
Un sistema multimodale è un sistema hardware/software che consente la ricezione, l'interpretazione, il processamento in input e la generazione in output di due o più modi interattivi in maniera integrata e coordinata. Le applicazioni multimodali dovrebbero essere in grado di adattarsi ai cambiamenti relativi alle capacità dei dispositivi, alle preferenze dell'utente e alle condizioni ambientali.

2.1.1 Multimodalità e disabilità visive

La multimodalità può migliorare l'accessibilità a servizi e informazioni, a utenti con disabilità visive. Uno dei problemi di questo tipo di utenti è il limitato campo visivo, inteso come perdita del contesto dell'applicazione. Questo è dovuto all'adozione di tecniche come l'ingrandimento del carattere e dello schermo, che permettono la visualizzazione di piccole quantità di informazioni alla volta. Alcuni eventi

dell'interfaccia non compresi nel loro campo visivo potrebbero non essere notificati. Fornendo output vocali aggiuntivi si potrebbe notificare all'utente le nuove informazioni o eventuali cambiamenti di stato.

Un altro problema può invece essere legato alla difficoltà nel riconoscimento visivo dei colori. Infatti molte persone non sono in grado di distinguere alcuni colori. Un principio da seguire è sicuramente quello di non utilizzare i colori come unica sorgente di informazione, ma in combinazione con testi, simboli o altro. Alcune combinazioni di colore di testo e sfondo possono produrre un testo illeggibile a determinati utenti. La soluzione è di fornire all'utente la possibilità di scegliere i colori che meglio si adattano alle sue capacità visive.

L'utilizzo di un'interfaccia vocale supplementare a quella visiva, è una delle soluzioni che ben si presta a migliorare l'accessibilità in presenza di questi tipi di disabilità. Output vocali possono infatti fare da guida nella navigazione dell'applicazione ed input vocali facilitare l'esecuzione delle funzioni dell'applicazione.

2.1.2 Linee guida per la progettazione di un'interfaccia multimodale

Un ruolo importante nello sviluppo delle applicazioni multimodali, è sicuramente ricoperto dalla progettazione dell'interfaccia utente. Allo scopo di ottenere un'interazione vicina a quella uomo -uomo e di aumentare la robustezza dell'interazione utilizzando informazioni complementari e ridondanti, sono state definite delle linee guida per la progettazione di interfacce multimodali (A.V. , *Guidelines for multimodal user interface design*. 2004) [16].

Possono essere individuate le seguenti categorie:

Specifica dei requisiti :

I punti critici nella progettazione di ogni applicazione sono i requisiti utente e le capacità di sistema del dominio selezionato. Alcune considerazioni sulla specifica dei requisiti di sistemi multimodali sono:

- Progettazione per un largo numero di utenti e contesti d'uso : chi progetta interfacce multimodali dovrebbe familiarizzare con caratteristiche psicologiche, livello d'esperienza, background culturale, caratteristiche fisiche (età, vista, udito), degli utenti. Un'applicazione sarà maggiormente accettata se potrà essere utilizzata da una larga popolazione è in più di una modalità.

- Sicurezza e privacy : in situazioni in cui l'utente desidera mantenere un determinato grado di sicurezza, interfacce che utilizzano riconoscimento vocale dovrebbero anche mettere a disposizione alternative non vocali. Lo stesso concetto vale per l'inserimento di dati sensibili come passwords o determinate informazioni personali.

Progettare input/output multimodale :

Per ottimizzare le prestazioni umane nei sistemi multimodali, alcuni principi possono essere utilizzati per la progettazione delle informazioni da presentare agli utenti, in particolar modo su come integrare differenti modalità o supportare input utente multipli (es. voce, gesti).

- Massimizzare capacità fisiche e cognitive : determinare come supportare interazioni intuitive ed efficienti, basate sulle capacità degli utenti :
    - Evitare di presentare informazioni in diverse modalità se non necessario; ad esempio se l'utente deve fare attenzione simultaneamente ad entrambe le sorgenti per comprendere il materiale presentato; tale ridondanza può incrementare il carico cognitivo a scapito della comprensione del materiale;
    - Massimizzare i vantaggi di ciascuna modalità in determinate situazioni e attività;
- Integrare le modalità in modo che siano compatibili con le preferenze dell'utente, il contesto e le funzionalità di sistema : quando si utilizzano modalità multiple, si dovrebbe tener conto dei seguenti principi :
    - Confrontare l'output con l'input accettabile, in modo che siano coerenti;
    - Utilizzare suggerimenti multimodali;
    - Assicurare la sincronizzazione temporale delle modalità di output;
    - Assicurare che lo stato corrente dell'interazione sia condiviso tra le diverse modalità e che l'appropriata informazione sia visualizzata.

Adattività:

Le interfacce multimodali dovrebbero adattarsi alle esigenze e capacità dei diversi utenti. Le differenze possono essere memorizzate in un profilo utente ed utilizzate per determinare le impostazioni dell'interfaccia; alcuni esempi sono :

- Permettere gesti per aumentare o sostituire l'input vocale in ambienti disturbati o per utenti con disabilità vocali;

- Adattare la quantità ed il metodo di presentazione delle informazioni sia all'utente che al dispositivo.

Consistenza:
Presentazioni e output vocali dovrebbero condividere le stesse caratteristiche nei limiti del possibile, e dovrebbero far riferimento alla stessa attività e la stessa terminologia. Ulteriori principi per la consistenza sono :

- Output indipendente dalle varie modalità di input
- Interazione di modalità combinate, attraverso le applicazioni
- Cambio di stato (es. modo d'interazione) iniziato dall'utente o dal sistema, assicurando il riconoscimento delle scelte dell'utente e notificando l'attività iniziata dal sistema.

Feedback all'utente:
Gli utenti dovrebbero essere resi consapevoli delle modalità d'interazione corrente e di quelle disponibili, senza però l'utilizzo di lunghe istruzioni. Alcuni esempi sono l 'utilizzo di icone descrittive (come microfoni per indicare pulsanti per il riconoscimento vocale) oppure la notifica all'utente dell'avvio della fase di riconoscimento vocale, se questa parte in automatico.

Gestione e prevenzione degli errori:
Gli errori utente possono essere diminuiti e la gestione degli errori migliorata fornendo chiari punti di uscita da un'attività, una modalità o dall'intero sistema, e la possibilità di annullare i comandi o le azioni precedenti. Altri specifici esempi sono:

- Dare all'utente il controllo sulla selezione delle modalità d'interazione
- Se si verifica un errore, dare la possibilità all'utente di cambiare modalità

2.2 SALT

Speech Application Language Tags (SALT) [17] è un linguaggio di markup per interfacce vocali. Consiste in un piccolo insieme di elementi XML, con associati attributi e proprietà, eventi, metodi di oggetti DOM (Document Object Model), che applicano un'interfaccia vocale alle pagine web.
SALT può essere usato con HTML, XHTML e altri standard per scrivere interfacce vocali per applicazioni puramente vocali (es. telefonia) e multimodali (interfaccia vocale + grafica). Comunque

SALT non estende un linguaggio di markup particolare, ma applica l'interfaccia vocale su un livello separato, in modo da essere utilizzabile con diversi linguaggi.

2.2.1 Architettura SALT

SALT può essere adottato su diversi dispositivi, quali PC, dispositivi mobili, telefoni. Ad esempio, i PC possono eseguire sia i processi di riconoscimento vocale che di output vocale in locale, mentre piccoli dispositivi come palmari con limitate capacità d'esecuzione potrebbero essere dotati di un browser compatibile SALT ma usare server remoti per il riconoscimento vocale e la sintesi vocale. Telefoni tradizionali potrebbero effettuare chiamate verso server dotati di browser puramente vocali compatibili SALT. Nella figura seguente si può vedere un'architettura di riferimento SALT di alto livello.

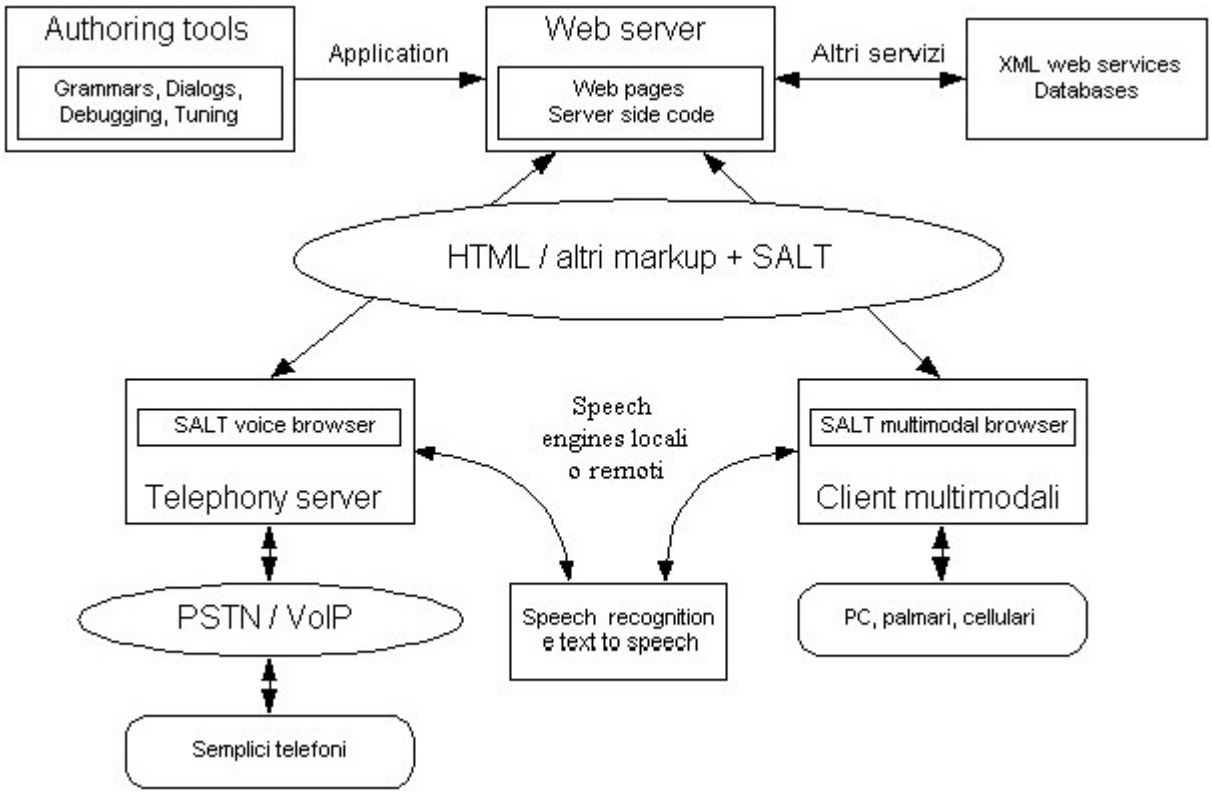

Figura 8 : Architettura di riferimento SALT

2.2.2 Elementi principali SALT

Esistono tre elementi principali di alto livello in SALT :

- <listen …> : configura il riconoscimento vocale, esegue riconoscimenti e gestisce gli eventi relativi all'input vocale;
- <prompt …> : configura il sintetizzatore ed esegue l'output vocale;
- <dtmf …> : configura e controlla DTMF nelle applicazioni telefoniche.

Elementi di input come <listen> e <dtmf> possono contenere i seguenti controlli :

- <grammar …> : per specificare le risorse relative alla grammatica da utilizzare per l'input
- <bind …> : per processare i risultati del riconoscimento vocale

L'elemento <listen> può inoltre contenere un controllo per la registrazione dell'input audio :

- <record …> : per la registrazione dell'input audio

<listen>

L'elemento *listen* è utilizzato sia per il riconoscimento vocale sia per la registrazione audio. Un elemento *listen* utilizzato per il riconoscimento vocale contiene una o più elementi *grammar*, che servono a specificare i possibili input dell'utente. Un elemento *listen* utilizzato per la registrazione audio contiene invece l'elemento *record*, utilizzato per configurare il processo di registrazione. In entrambi i casi l'elemento *bind* può essere utilizzato per processare i risultati del riconoscimento o della registrazione.

Un semplice esempio di *listen* che contiene una grammatica remota per nomi di città, ed un elemento bind per processare il risultato del riconoscimento vocale è il seguente :

```
<salt:listen id="travel">
   <salt:grammar src="./city.xml" />
   <salt:bind targetElement="txtBoxOriginCity"
        value="/result/origin_city" />
</salt:listen>
```

L'elemento *listen* è inoltre fornito di :

- attributi : per la configurazione del riconoscimento vocale (periodi di timeout, riconoscimento singolo o multiplo, ecc.);
- proprietà : contengono i risultati ottenuti dal processo di riconoscimento vocale;

- metodi : servono al controllo del processo di riconoscimento; con questi metodi è possibile iniziare, finire, cancellare un processo di riconoscimento, attivare e disattivare regole grammaticali di alto livello.

Inoltre sono supportati diversi eventi relativi al processo di riconoscimento, che possono essere gestiti tramite la definizione di attributi del tag <listen …>.

<grammar>

L'elemento *grammar* è utilizzato per specificare grammatiche, ed è un elemento figlio di *listen*. Per il riconoscimento vocale deve essere specificata almeno una grammatica, in una delle due modalità possibili : direttamente nella pagina, oppure tramite riferimento ad un file tramite l'attributo *src*. SALT è indipendente dai formati utilizzati per la definizione delle grammatiche, comunque i browser SALT devono almeno supportare il formato XML di W3C Speech Recognition Grammar Specification (per ulteriori informazioni http://www.w3.org/TR/speech-grammar/) per garantire l'interoperabilità.

Di seguito è mostrato un esempio di utilizzo dell'elemento *grammar* : per riferimento e per esplicita definizione all'interno del tag (seguendo W3C SRGS grammar). Lo scopo della grammatica dell'esempio è quello si definire i nomi delle città che possono essere riconosciuti durante il processo di riconoscimento.

```
<salt:grammar src="cities.grxml" type="application/srgs+xml" />
              or
<salt:grammar xmlns="http://www.w3.org/2001/06/grammar">
   <salt:grammar root="root">
      <rule id="root">
         <item repeat="0-1">from
         </item>
         <ruleref name="#cities" />
      </rule>
      <rule id="cities">
          <one-of>
             <item> Cambridge </item>
             <item> Seattle </item>
             <item> London </item>
          </one-of>
      </rule>
   </salt:grammar>
```

```
</salt:grammar>
```

<bind>

L'elemento *bind* è utilizzato per collegare i valori provenienti dall'input vocale ad elementi della pagina, ma anche per lanciare metodi di elementi della pagina. Il risultato dell'input processato dall'elemento *bind* è un documento XML contenente semantic markup language (W3C Natural Language Semantic Markup Language) per la specifica dei risultati del riconoscimento. Il suo contenuto è solitamente composto da valori semantici, parole dettate dall'utente e punteggi di confidenza.

Per assicurare l'interoperabilità, i browser SALT devono supportare formato W3C Recommendation for Natural Language Semantic Markup Language (NLSML) (per ulteriori informazioni http://www.w3.org/TR/nl-spec/).

Un esempio di risultato WRC NLSML per l'espressione "I'd like to travel from Seattle to Boston" è :

```
<result grammar="http://flight" xmlns:xf="http://www.w3.org/2000/xforms">
   <interpretation confidence="0.4">
     <input mode="speech">
       I'd like to travel from Seattle to Boston
     </input>
     <xf:instance>
       <airline>
        <origin_city confidence="0.45">Seattle</origin_city>
        <dest_city confidence="0.35">Boston</dest_city>
       </airline>
     </xf:instance>
   </interpretation>
</result>
```

I valori che devono essere collegati con gli elementi della pagina, sono riferiti utilizzando una query XPath. Gli elementi della pagina sono identificati tramite l'attributo *targetelement* dell'elemento *bind*.

Partendo dal documento XML dell'esempio sopra (risultato di un riconoscimento vocale), l'elemento listen seguente, utilizza l'elemento bind per trasferire i valori origin_city e dest_city nei textbox della pagina txtBoxOrigin e txtBoxDest.

```
<html xmlns:salt="http://www.saltforum.org/2002/SALT">
   ...
   <form id="formTravel">
      <input name="txtBoxOrigin" type="text"/>
      <input name="txtBoxDest" type="text" />
   </form>
   ...
   <salt:listen id="listenTravel">
      <salt:grammar src="./city.grxml" />
      <salt:bind targetelement="txtBoxOrigin"
            value="//origin_city" />
      <salt:bind targetelement="txtBoxDest"
            value="//dest_city" />
   </salt:listen>
   ...
</html>
```

<record>

La registrazione vocale è abilitata su un elemento *listen* utilizzando l'elemento *record*. La registrazione vocale può essere utilizzata al posto del riconoscimento vocale o in aggiunta ad esso. Sostanzialmente questo elemento permette di configurare i parametri audio e altre informazioni relative alla registrazione dell'input vocale.

<prompt>

L'elemento *prompt* è utilizzato per specificare il contenuto dell'output audio. Il contenuto potrebbe essere di uno dei seguenti tipi :

- testo;
- valori contenuti in elementi del documento (es. value di un textbox) ;
- link a file audio.

Un semplice elemento *prompt* necessita della definizione del testo richiesto per l'output :

```
<salt:prompt id="welcome">
      Welcome to the home page
</salt:prompt>
```

SALT permette l'utilizzo di un qualsiasi formato di speech synthesis markup language all'interno dell'elemento *prompt*. Per assicurare interoperabilità, i browser SALT devono supportare W3C Recommendation for Speech Synthesis Markup Language (SSML) (per ulteriori informazioni http://www.w3.org/TR/speech-synthesis/). Per enfatizzare alcune parti di una frase si potrebbe utilizzare SSML come segue :

```
<salt:prompt id="welcome" xmlns:ssml="http://www.w3.org/2001/10/synthesis">
      Welcome to the <ssml:emphasis> home page </ssml:emphasis>
</salt:prompt>
```

L'elemento *value*, figlio dell'elemento *prompt*, può essere utilizzato per ottenere i valori di un elemento del documento. Attraverso gli attributi targetelement e targetattribute, si può specificare quale elemento e attributo si vuole ottenere. Ad esempio per ottenere gli attributi value dei textbox txtBoxOrigin e txtBoxDest si può agire nel seguente modo :

```
<salt:prompt id="confirm">
      Do you want to travel from
      <value targetelement="txtBoxOrigin" targetattribute="value">
      to
     <value targetelement="txtBoxDest" targetattribute="value">
     ?
</salt:prompt>
```

L'elemento *content*, invece può essere utilizzato per riferire contenuti esterni come speech markup generati dinamicamente o file audio remoti. Questo elemento sostanzialmente specifica un link ad una risorsa esterna e ne identifica il tipo. L'esempio seguente mostra un possibile utilizzo di *content*, per un contenuto XML in SSML e per un file audio.

```
<salt:prompt>
   <salt:content href="/VoiceMailWelcome.ssml" type="application/ssml+xml">
   After the beep, please record your message
   <salt:content href="/wav/beep.wav">
</salt:prompt>
```

L'elemento *prompt* è inoltre fornito di :

- attributi : per la configurazione dell'output audio (possibilità di stoppare l'esecuzione, scelta della lingua, ecc.);
- proprietà : contengono informazioni e stato del processo di esecuzione audio;
- metodi : servono per il controllo dell'esecuzione del *prompt*; con questi metodi si può far partire l'esecuzione di un *prompt* o accodarla in una coda di *prompt*.

Inoltre sono supportati diversi eventi relativi al processo di esecuzione, che possono essere gestiti tramite la definizione di attributi del tag <prompt ..>.

<dtmf>

L'elemento *dtmf* è utilizzato nelle applicazioni telefoniche per definire grammatiche DTMF e gestire l'input da tastiera telefonica. Come l'elemento *listen,* i suoi principali elementi figlio sono *grammar* e *bind*. Contiene inoltre risorse per la configurazione del processo di collezione DTMF e la gestione dell'input da tastiera e dei timeouts. Come *listen* è dotato di metodi, tra i quali start e stop per l'esecuzione e terminazione del processo.

L'esempio seguente mostra come ottenere un numero telefonico.

```
<salt:dtmf id="dtmfPhoneNumber">
     <salt:grammar src="7digits.gram" />
     <salt:bind value="/result/phoneNumber" targetElement="iptPhoneNumber" />
</salt:dtmf>
```

2.3 VoiceXML

Voice eXtensible Markup Language (VoiceXML) [19], è stato progettato per la realizzazione di interfacce vocali in grado di supportare : sintetizzazione audio, audio digitalizzato, riconoscimento vocale, input DTMF, registrazione audio, telefonia. Uno degli scopi principali è quello di portare i vantaggi dello sviluppo web e della distribuzione dei contenuti, alle applicazioni vocali interattive tradizionali.

2.3.1 Architettura VoiceXML

Il modello architetturale è rappresentato nella figura seguente :

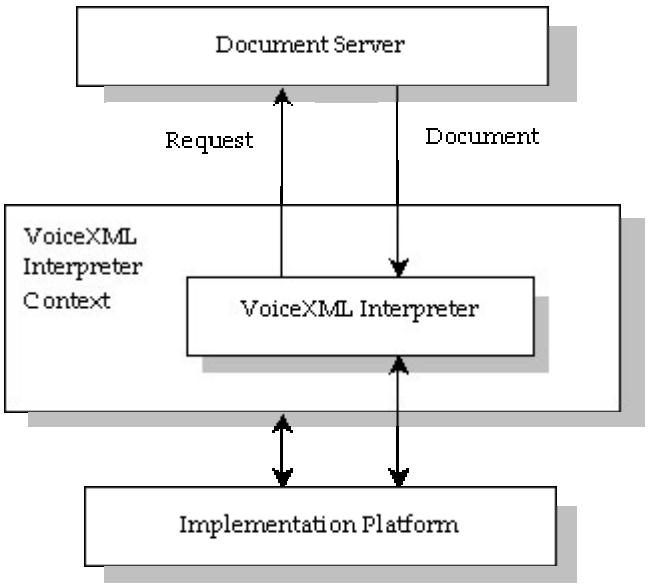

Figura 9 : Modello architetturale VoiceXML

Un *Document Server* (che potrebbe essere un Web Server) processa le richieste di un'applicazione client, il *VoiceXML Interpreter*, attraverso il *VoiceXML Interpreter Context*. Il server produce in risposta documenti VoiceXML (documenti XML), che sono processati dal *VoiceXML Interpreter*. L'*Implementation Platfrom* è controllata dal *VoiceXML Iterpreter Context* e dal *VoiceXML Interpreter*. Per esempio in un'applicazione vocale interattiva il *VoiceXML Interpreter Context* potrebbe occuparsi di riconoscere la chiamata entrante, acquisire il documento VoiceXML iniziale e rispondere alla chiamata, mentre il *VoiceXML Interpreter* condurrebbe il dialogo dopo la risposta seguendo le specifiche del documento VoiceXML.

VoiceXML è utilizzato prevalentemente per applicazioni puramente vocali (applicazioni telefoniche) ma può anche essere utilizzato per applicazioni multimodali (grafica+voce), come può essere visto nella figura seguente :

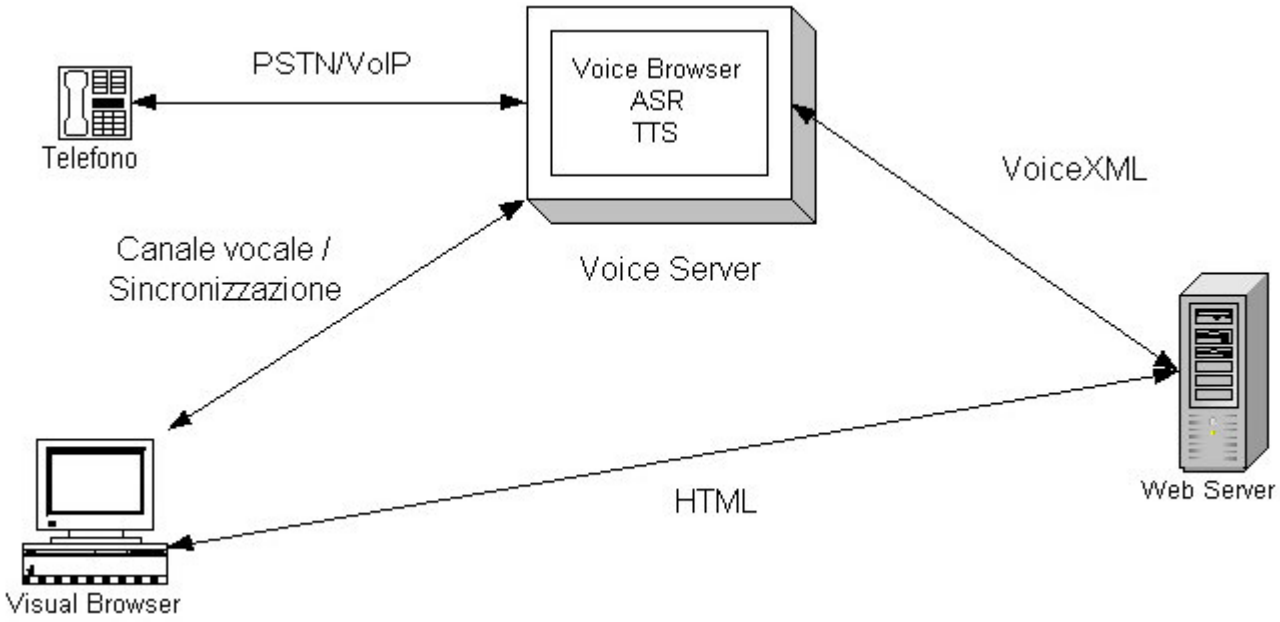

Figura 10 : Architettura implementativa VoiceXML

Il Voice Server permette la comunicazione vocale implementando un Voice Browser in grado di interpretare i documenti VoiceXML; all'interno del Voice Browser sono implementati il Text To Speech (TTS) per la traduzione da testo a voce, e l'Automatic Speech Recognition (ASR) per il riconoscimento dell'input vocale.

Per l'accesso multimodale, è necessaria una sincronizzazione tra i due browser, attraverso l'implementazione di agent di sincronizzazione, uno per browser, per la gestione dei link a documenti vocali e la notifica di eventi.

Una soluzione alternativa per l'accesso multimodale è costituita dall'utilizzo di un unico browser multimodale in grado di interpretare anche elementi VoiceXML. Il browser Opera è un esempio di browser multimodale compatibile con le specifiche XHTML + VoiceXML.

2.3.2 Elementi principali VoiceXML

Un documento VoiceXML (o un insieme di documenti chiamato *application*) è paragonabile ad una macchina a stati finiti. Ogni documento è formato da elementi di alto livello chiamati *dialog*. L'utente si trova sempre in un determinato stato, o dialog, all'interno di una conversazione. Le transizioni sono specificate da URI, che definiscono il prossimo documento e dialog da utilizzare. Se non è indicato

alcun documento, si assume che sia il documento corrente. L'esecuzione termina quando un di alog non specifica il successore o contiene un elemento che termina la conversazione.
Un esempio di documento VoiceXML è il seguente :

```
<?xml version="1.0"?>
<vxml version="1.0">
  <form>
    <block>Hello World!</block>
  </form>
</vxml>
```

Questo esempio sintetizza e presenta la frase "Hello World!" all'utente.
L'elemento di top-level è <vxml>, e sostanzialmente è un contenitore di dialog. Ci possono essere due tipi di dialog : *forms* e *menus*.
L'elemento form può contenere i seguenti elementi :

- Un insieme di elementi che sono valutati nel ciclo principale dell'algoritmo di interpretazione form; sono suddivisi in *field*, che definiscono le variabili relative ai campi del form, e *control,* che aiutano a controllare il processo di inserimento valori nei campi del form;
- Dichiarazione di variabili diverse da field;
- Gestori di eventi;
- Blocchi di istruzioni che sono eseguite in seguito all'avvenuto riempimento di determinati field.

L'esempio seguente presenta un possibile servizio meteorol ogico :

```
<form id="weather_info">
   <block>Welcome to the weather information service.</block>
   <field name="state">
     <prompt>What state?</prompt>
     <grammar src="state.gram" type="application/x-jsgf"/>
     <catch event="help">
       Please speak the state for which you want the weather.
     </catch>
   </field>
   <field name="city">
     <prompt>What city?</prompt>
     <grammar src="city.gram" type="application/x-jsgf"/>
```

```
        <catch event="help">
          Please speak the city for which you want the weather.
        </catch>
      </field>
      <block>
         <submit next="/servlet/weather" namelist="city state"/>
      </block>
    </form>
```

Un possibile dialogo sarebbe :

C (computer): Welcome to the weather information service. What state?
H (human): Help
C: Please speak the state for which you want the weather.
H: Georgia
C: What city?
H: Macon
C: The conditions in Macon Georgia are sunny and clear at 11 AM …

Ogni elemento field nell'esempio ha un'elemento *prompt* per interrogare l'utente, un'elemento *grammar* per definire le possibili risposte, e un gestore dell'evento *help*. La risposta dell'utente è utilizzata per valorizzare la variabile relativa all'elemento field (state,city).

L'elemento menu permette all'utente di effettuare una scelta tra diverse opzioni, e passare al dialog relativo alla scelta. Il seguente menu offre all'utente tre opzioni :

```
<menu>
  <prompt>Welcome home. Say one of: <enumerate/></prompt>
  <choice next="http://www.sports.example/vxml/start.vxml">
    Sports </choice>
  <choice next="http://www.weather.example/intro.vxml">
    Weather </choice>
  <choice next="http://www.stargazer.example/voice/astronews.vxml">
    Stargazer astrophysics news </choice>
  <noinput>Please say one of <enumerate/></noinput>
```

```
</menu>
```

Un possible dialogo sarebbe:

C: Welcome home. Say one of: sports; Weather; Stargazer astrophysics news.

H: sports.

C: *(proceeds to http://www.sports.example/vxml/start.vxml)*

L'elemento *choice* permette di definire gli elementi grammaticali relativi alle scelte, e l'URI relativo al documento VoiceXML relativo alla scelta. L'elemento *enumerate* permette di generare automaticamente un descrizione delle possibili scelte, considerando gli elementi choice.

3 IL CONTESTO APPLICATIVO E OBIETTIVI DELL'APPLICAZIONE

3.1 Analisi della piattaforma Lezi.NET

In questo paragrafo sarà analizzata la piattaforma Lezi.NET su cui dovrà essere integrato MultiLezi.NET. In particolare saranno prese in considerazione le funzionalità e l'architettura.

3.1.1 Funzionalità generali

Le principali funzionalità messe a disposizione da Lezi.NET sono :

- Fruizione corso : interfaccia web che permette la fruizione di un corso IMS-SCORM, mettendo a disposizione la navigazione dei contenuti e la loro visualizzazione;
- Gestione corso : pagina che mette a disposizione informazioni sul corso, lista di azioni da eseguire sul corso, ACL (Access Control List) del corso;
- Gestione progetti : editor che permette la creazione e l'eventuale modifica di corsi;
- Gestione risorse : pagina che simula il comportamento di un file manager, per l'org anizzazione dei file delle risorse:
- Gestione iscrizioni : pagina che permette di visualizzare e processare le richieste di sottoscrizione a corsi.

3.1.2 Architettura

L'architettura della piattaforma Lezi.NET ha una struttura multi -tier come è possibile vedere nella figura seguente :

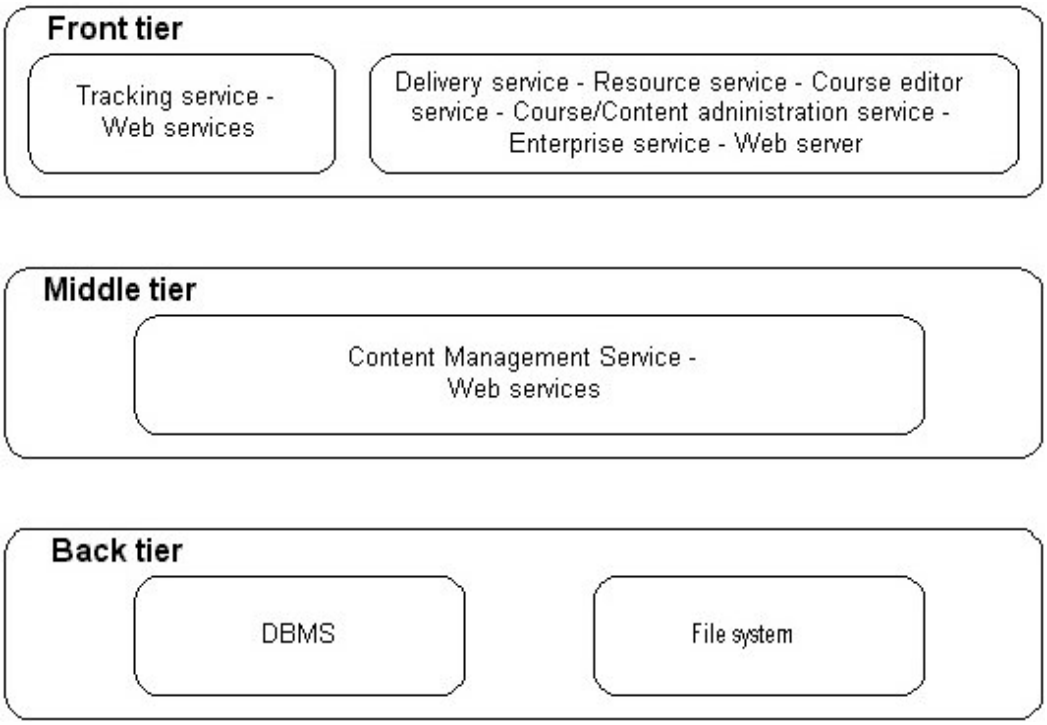

Figura 11 : Architettura multi-tier di Lezi.NET

- Back tier : ospita un DBMS e servizi per la memorizzazione di dati su file system. I package dei corsi e le risorse caricate dagli utenti sono memorizzati su file system ed indicizzati da un riferimento presente nel database. E' stata effettuata questa scelta per questioni di performance : notoriamente l'accesso a file system è più veloce rispetto a quello a database e non impone restrizioni di dimensione. I vari file xml utilizzati dall'applicazione sono invece memorizzati direttamente nel database in campi BLOB.
- Middle tier : offre servizi al front tier occupandosi degli aspetti relativi alla memorizzazione e gestione delle risorse, dei corsi, dei progetti e di tutti gli altri oggetti del sistema. Si tratta di un Content Management Service (CMS) implementato tramite Web Services. I principali servizi messi a disposizione sono :
    - Creazione, cancellazione, modifica di utenti e gruppi;
    - Creazione, cancellazione, modifica di proprietà, pubblicazione, deployment, download di corsi;
    - Creazione, cancellazione, apertura, chiusura, modifica di progetti di corsi;
    - Pubblicazione e gestione news e commenti;

  - Trasferimento, cancellazione, modifica di proprietà di files di risorse archiviati;
  - Archiviazione dei dati provenienti dal tracking service riguardanti l'interazione con l'utente durante la fruizione dei corsi;
  - Memorizzazione e gestione delle Access Control List delle diverse entità di sistema;
  - Autenticazione degli utenti e autorizzazione all'esecuzione delle diverse operazioni.
- Front tier : fornisce la parte di presentazione dell'applicazione ed il tracking service; la presentazione è implementata mediante pagine aspx (ASP.NET), mentre il tracking service è implementato tramite Web Services e fornisce la parte server-side del run-time environment SCORM.

La comunicazione tra middle tier e front tier avviene mediante protocollo http su porta 80, garantendo un maggior livello di sicurezza; è possibile infatti porre un firewall tra i due tier. Anche il front tier può essere protetto da firewall in quanto il protocollo di comunicazione è sempre http su porta 80, sia che si tratti di Web Services che di altri servizi (http + html). La figura seguente descrive l'architettura Lezi.NET in maniera dettagliata, evidenziando le scelte che riguardano la piattaforma software.

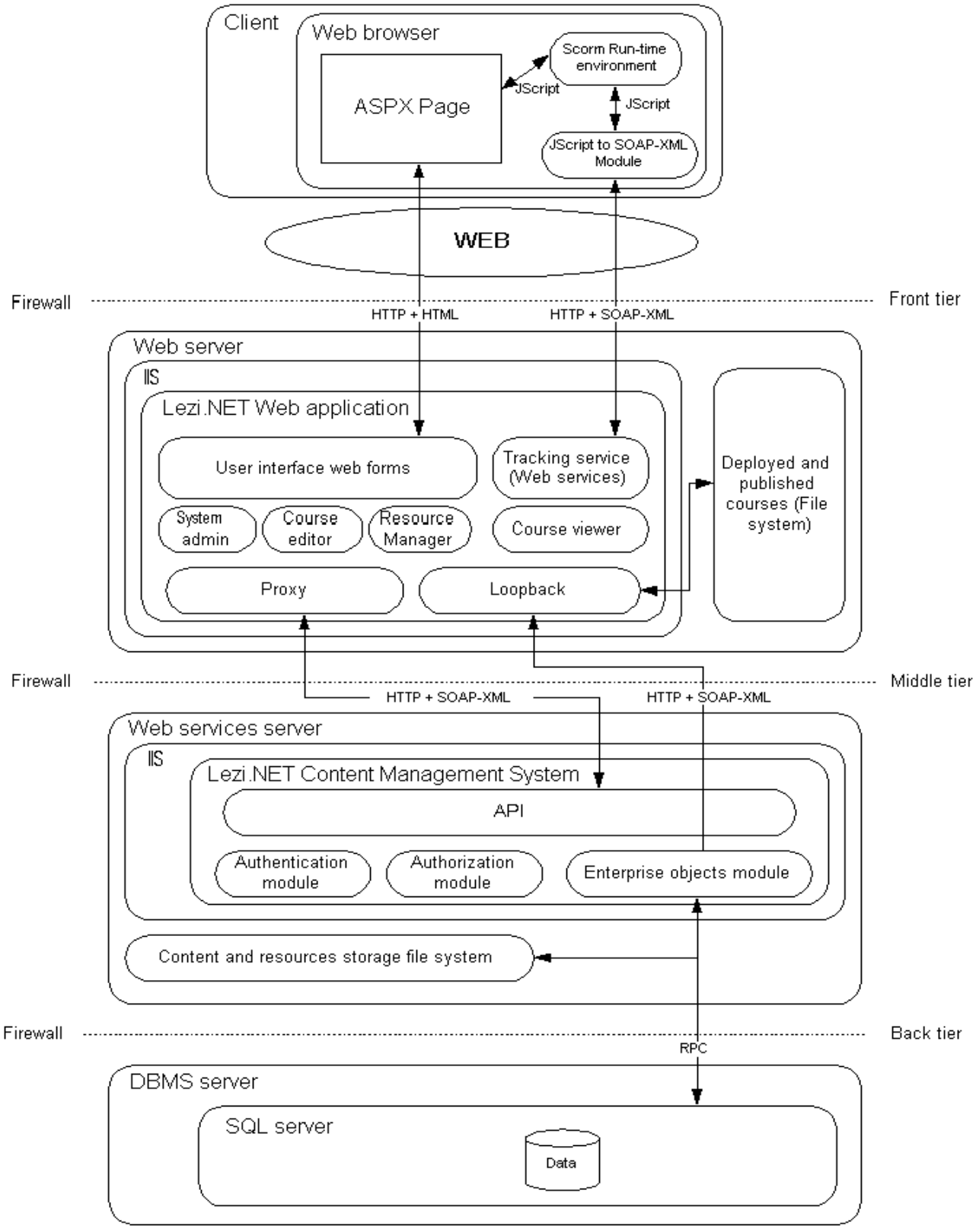

Figura 12 : Architettura dettagliata Lezi.NET

3.2 Obiettivi dell'applicazione

L'obiettivo principale è quello di realizzare un servizio per utenti ipovedenti in un ambiente di e-learning. L'utente tipo ha notevoli difficoltà visive, ma è in grado di interagire manualmente, attraverso l'utilizzo della tastiera e del mouse.
L'intento è quello di permettere l'accesso alla piattaform a di e-learning e la fruizione di corsi, adottando tecniche audio e visive appropriate, in grado di guidare l'utente. In quest'ottica l'utilizzo di un'interfaccia multimodale risulta fondamentale : infatti integrando grafica e interazione vocale si pensa di mettere a disposizione dell'utente un servizio in grado di supportare le sue difficoltà visive.
Per realizzare questo servizio si pensa di utilizzare come base di partenza una piattaforma di e-learning già esistente, Lezi.NET, e di sviluppare un ulteriore canale di accesso dotato delle caratteristiche sopra descritte. Gli accessi alla piattaforma devono comunque essere sincronizzati al fine di tenere traccia dei percorsi formativi degli utenti indipendentemente dal canale utilizzato. Ad esempio, il completamento della lezione di un corso effettuata su un canale deve risultare anche sull'altro.
Un'altro obiettivo è quello di analizzare le principali tecnologie disponibili per il *text to speech* (da testo a discorso) e lo *speech to text* (da discorso a testo) nell'ottica di effettuare una scelta per lo sviluppo dell'applicazione. Si vuole inoltre testare la reale efficacia dell'utilizzo di queste tecnologie per lo speech in applicazioni multimodali. Le tecnologie candidate sono VoiceXML e SALT; saranno messe a confronto per individuare quella più adatta allo scenario delineato dagli obiettivi prefissi.

3.3 VoiceXML vs SALT

VoiceXML e SALT sono entrambi linguaggi di markup per la definizione di interfacce vocali, anche se operano in modi differenti.
VoiceXML è orientato alle applicazioni telefoniche, ed in particolare è stato sviluppato per creare applicazioni IVR (Interactive Voice Response) utilizzando un linguaggio di markup e beneficiando così dei vantaggi del World Wide Web. Permette la definizione di dialoghi vocali per telefoni e cellulari.
SALT invece punta ad applicazioni vocali su diversi dispositivi, quali telefoni, PDA, PC. Il supporto per interazioni multimodali è una caratteristica di SALT. Permette la definizione di dialogi vocali e applicazioni multimodali.
Le principali differenze tecniche sono illustrate di seguito :

- Raggio d'azione : VoiceXML include interfaccia vocale, modello dati, e strutture per il controllo del flusso mentre SALT è focalizzato sull'interfaccia vocale.
    - VoiceXML mette a disposizione un alto numero di tag poiché mette a disposizione modelli dati (es. <form>, <menu>) e programmazione procedurale (es. <if>, <then>, <goto>) oltre all'interfaccia vocale (es. <prompt>, <grammar>). Questa scelta è stata fatta al fine di mettere a disposizione un linguaggio di markup standalone per la descrizione di tipici dialoghi telefonici.
    - SALT mette a disposizione pochi tag per la definizione dell'interfaccia vocale (es. <listen>, <prompt>). SALT è stato pensato per poter lavorare su diversi dispositivi e ambienti di esecuzione web. SALT non definisce un modello di esecuzione, ma utilizza i modelli di esecuzione web standard (es. HTML+ECMAScript, WML), permettendo di sviluppare applicazioni multimodali su diversi dispositivi.
- Modello di programmazione : VoiceXML contiene un algoritmo per l'esecuzione degli elementi predefinito, mentre SALT permette agli sviluppatori di creare flussi di dialogo personalizzati.
    - VoiceXML contiene un algoritmo che gestisce il flusso per la valorizzazione dei campi (<field>) dei vari form (<form>). L'algoritmo può essere manipolato con l'utilizzo di elementi condizionali e procedurali di programmazione, col fine di creare dialoghi complessi.
    - SALT non contiene un algoritmo d'esecuzione predefinito, ma utilizza il modello a eventi. In questo modello, output e input vocali sono attivati da eventi della pagina web. Questi eventi possono essere scatenati dall'interfaccia grafica (es. click di un bottone), dai dati (es. cambio di un valore in un campo) oppure da altri elementi SALT (es. un mancato riconoscimento vocale). Gli elementi SALT espongono un'interfaccia composta da attributi, proprietà, metodi e gestori di eventi, che è completamente integrata nel modello di esecuzione della pagina.
- Modello architetturale : VoiceXML necessita di un voice interpreter, SALT si adatta ai tradizionali modelli web.
    - VoiceXML è composto da documenti XML che stanno solitamente su un web server e devono essere interpretati da un voice interpreter presente su un voice server (vedi 4.2 VoiceXML).
    - SALT è integrato nelle normali pagine web, ed i suoi elementi sono interpretati da browser web abilitati SALT (vedi 4.1 SALT).

Sia VoiceXML che SALT utilizzano standard W3C : SRGS per il formato della grammatica (http://www.w3.org/TR/speech-grammar), SSML per il formato dell'output vocale (http://www.w3.org/TR/speech-synthesis), NLSML per il formato dei risultati dei riconoscimenti vocali (http://www.w3.org/TR/nl-spec), CCXML per il controllo delle chiamate telefoniche (http://www.w3.org/TR/ccxml).
Anche se VoiceXML è nato soprattutto per applicazioni puramente vocali, come quelle telefoniche, è possibile implementare applicazioni multimodali principalmente in due modi : utilizzando un browser web ed un browser vocale in grado di sincronizzarsi mediante l'impiego di appositi agent, oppure integrando codice VoiceXML in pagine web seguendo il modello SALT.

3.4 Scelta tecnologica

La scelta tecnologica per l'interfaccia vocale è stata presa tenendo presente gli obiettivi prefissi. Soprattutto il requisito della multimodalità ha fatto si che venisse selezionata la tecnologia SALT. Infatti SALT offre un supporto semplice e familiare allo sviluppatore web, per lo sviluppo di applicazioni multimodali; VoiceXML è più orientato allo sviluppo di applicazioni puramente vocali, anche se è possibile sviluppare applicazioni multimodali, introducendo però maggiore complessità di sviluppo rispetto a SALT. Utilizzando invece XHTML+VoiceXML si è limitati dall'utilizzo dei pochi browser in grado di interpretare queste specifiche (es. il browser Opera).
Un altro aspetto è costituito dalle tecnologie utilizzate per lo sviluppo della piattaforma Lezi.NET; si tratta prevalentemente di tecnologie Microsoft quali .NET Framework [13], ASP.NET [12], C# [3]. L'esistenza di un kit di sviluppo per applicazioni vocali della Microsoft, basato sulla tec nologia SALT, ha influito notevolmente sulla scelta. Si tratta di Speech Application SDK, un kit completamente integrabile nell'ambiente di sviluppo Visual Studio .NET 2003.

3.4.1 Speech Application SDK

Speech Application SDK [18] è un kit per sviluppatori che permette di aggiungere interfacce vocali ad applicazioni web ASP.NET. SASDK contiene diverse risorse, tra le quali controlli ASP.NET per l'interfacciamento vocale, una ricca libreria di grammatiche, tool di sviluppo per dialoghi; tutto ciò permette agli sviluppatori di creare applicazioni che riconoscono comandi vocali e comunicano risposte vocali o visuali. Possono essere create applicazioni di due tipi :

- Puramente vocali : l'utente interagisce vocalmente con una applicazione web ASP.NET, attraverso una connessione telefonica.
- Multimodali : l'utente interagisce vocalmente e manualmente con una applicazione web ASP.NET, in esecuzione su un browser visuale di un PC o dispositivo mobile.

SASDK si basa sullo standard per lo speech SALT; in particolare SASDK mette a disposizione dei controlli ASP.NET in grado di generare codice HTML+SALT. Per l'interpretazione dei tag SALT, SASDK mette a disposizione un add-in per Microsoft Internet Explorer.

I principali componenti di SASDK sono :

- Speech Controls : mettono a disposizione api di alto livello rispetto a SALT; in una pagina web costruita con Speech Controls, il web server traduce ogni tag relativo a Speech Controls in codice SALT eseguibile lato client, corredato da script necessari all'integrazione nell'appl icazione. Quindi il browser riceve HTML, SALT e script.
- Tools di sviluppo per Visual Studio .NET 2003 : si tratta di un editor per grammatiche, un editor per prompt, ed un editor per il controllo del dialogo.
- Add-in per Microsoft Internet Explorer : questo add-in esegue il riconoscimento vocale, l'output vocale e scambia HTML, SALT e script con il web server.
- Add-in per Microsoft Pocket Internet Explorer : add-in per la versione pocket di Internet Explorer; a differenza della versione per PC, il riconoscimento vocale e l'output vocale sono eseguiti lato server.

Per le applicazioni multimodali costruite per essere eseguite su PC è sufficiente il SASDK, ma per applicazioni destinate a Pocket PC e telefoni occorre un componente lato server, il Microsoft Speech Server (MSS). MSS mette a disposizione i seguenti servizi :

- Speech Engine Services (SES) : mette a disposizione i servizi di riconoscimento e output vocale per client telefonici e Pocket PC; SES riceve input vocale e restituisce i risultati del riconoscimento, riceve testo in formato markup e produce output vocale.
- Telephony Application Services (TAS) : in una applicazione puramente vocale, TAS interpreta le pagine abilitate SALT fungendo da client; TAS si interfaccia con hardware e software per le connessioni telefoniche; sostanzialmente gestisce la trasmissione dell'audio e del markup al SES, e permette l'interazione attraverso la tastiera telefonica.
- Telephony Interface Manager (TIM) : permette la comunicazione tra piattaforma telefonica installata e Microsoft Speech Server.

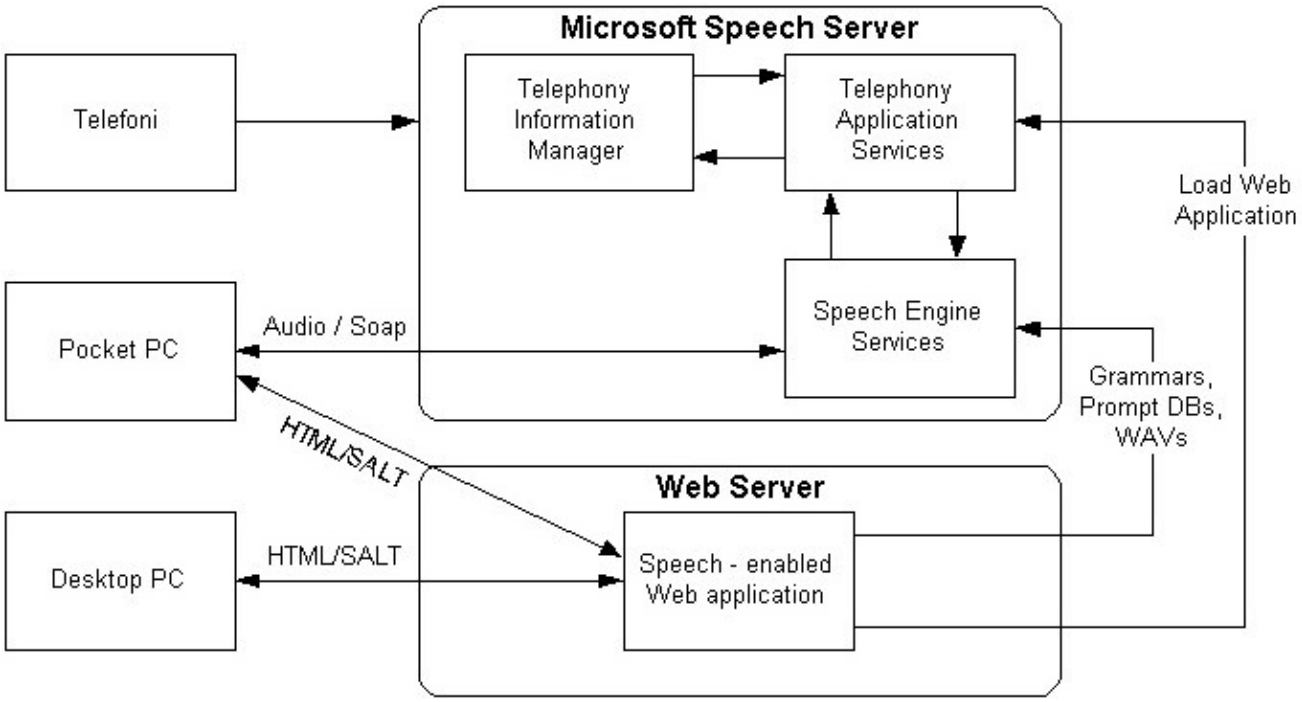

Figura 13: Parti principali di una soluzione speech, utilizzando la piattaforma speech di Microsoft

Ci sono 3 gruppi principali di Speech Controls :

- Basic speech controls :

  Sono i controlli che implementano gli elementi base SALT <prompt> e <listen>.

  - Prompt : permette di produrre output vocale a partire da testo (text to speech) o da registrazioni vocali presenti in un Prompt Database. Il testo può essere fisso o relativo a qualche elemento della pagina (es. testo contenuto in un textbox).
  - Listen : permette di ricevere l'input vocale. Ad esso possono essere collegate delle grammatiche (Grammar) per il riconoscimento semantico dell'input, e possono essere definiti dei collegamenti ad elementi della pagina (es. inserire l'input in un textbox).

- Dialog speech controls :

  Hanno lo scopo di controllare il flusso del discorso e gestire i dati del discorso. Questi controlli sono solitamente utilizzati per applicazioni vocali in cui non si ha una GUI, oppure in applicazioni dotate di GUI in cui non si ha la possibilità di interagire manualmente. Esiste uno script lato-client chiamato RunSpeech che gestisce ed attiva questi controlli.

  I principali controlli sono :

- Semantic Item : memorizzano informazioni relative ad un singolo elemento semantico, che può essere riconosciuto dall'input vocale dell'utente; il controllo può essere collegato ad un elemento target della pagina (esempio textbox per ricevere il testo riconosciuto).
- Semantic Map : è una collezione di Semantic Item e serve a memorizzare tutti i dati riconosciuti dall'input vocale dell'utente.
- QA : implementa una singola interazione del tipo domanda risposta; utilizza altri controlli per poter effettuare le operazioni, quali Prompt (per effettuare la domanda, confermare una risposta) e Listen (per ricevere la risposta); il risultato del riconoscimento vocale può essere memorizzato in Semantic Item, ma devono essere impostati gli elementi Grammar per il riconoscimento dell'input vocale.
- CompareValidator : per confrontare l'input dell'utente con qualche altro elemento della pagina o costante. Per ottenere l'input questo controllo fa riferimento ad un Semantic Item. In base al risultato del confronto possono essere configurati dei Prompt di risposta.
- CustomValidator : come il CompareValidator, ma il confronto viene effettuato da uno script definito dallo sviluppatore.
- Command : per implementare comandi vocali come "help", "cancel"; la modalità di riconoscimento implica la definizione di un Grammar.
- Grammar : configura la grammatica e le regole della grammatica da utilizzare; la grammatica ha formato xml e serve a definire il testo (parole o frasi) che può essere riconosciuto da un input vocale; infatti a partire dalla grammatica viene generato SML (Semantic Markup Language) che contiene le informazioni relative al testo riconosciuto, in modo da poter essere utilizzate dagli altri controlli (Es. Semantic Item).
- SpeechControlSettingItems : per specificare le proprietà di un certo gruppo di controlli.
- SpeechControlSettings : insieme di SpeechControlSettingItems.

- Application speech control :

  Si tratta di controlli che permettono l'implementazione di funzionalità di uso comune, quali il riconoscimento di numeri, date ecc. Sostanzialmente sono composizioni di Dialog speech controls , grammatiche predefinite e Prompt. Alcuni di questi controlli sono :

  - DataTableNavigator : per la navigazione vocale di una tabella di dati (next, read, ecc.).
  - ListSelector : per la selezione vocale di dati, da una listbox.
  - AlphaDigit : per riconoscere una stringa di numeri e lettere da input vocale.

- CreditCardDate : per riconoscere mese ed anno di una data (scadenza carta di credito) da input vocale.
- CreditCardNumber : per riconoscere un numero di carta di credito e tipo da input vocale.
- Currency : per riconoscere una somma in dollari da input vocale.
- Date : per riconoscere una data , dettata per intero o in sequenza (giorno,mese,anno) da input vocale.
- NaturalNumber : per riconoscere un numero naturale da input vocale.
- Phone : per riconoscere un numero di telefono nel formato U.S. , da input vocale.
- SocialSecurityNumber : per riconoscere un social security number da input vocale.
- YesNo : per riconoscere una risposta Yes o No da input vocale.
- ZipCode : per riconoscere un codice postale nel formato U.S. , da input vocale.

4 ANALISI E PROGETTAZIONE DELL'APPLICAZIONE MULTILEZI.NET

4.1 Analisi dei requisiti

L'applicazione che si vuole sviluppare, come già specificato nell'introduzione, è un'estensione dell'applicazione Lezi.NET alla multimodalità. Nello specifico le funzioni di Lezi.NET che si vogliono implementare sono la consultazione e fruizione dei corsi. Si vuole fornire un servizio adattativo con particolare attenzione al supporto a disabilità sensoriali. Nello specifico si tratta di disabilità visive tipiche di uno studente ipovedente. Occorre quindi tenere presente dei requisiti generali :

- Amplificazione/contrazione : nell'ottica dell'adattabilità all'utente, l'applicazione deve permettere la modifica di alcuni parametri grafici :
    - Dimensione : possibilità di ingrandire o rimpicciolire il carattere; l'utente deve poter scegliere la misura che gli permette la migliore visualizzazione possibile;
    - Contrasto : possibilità di cambiare il contrasto tra colore del testo e dello sfondo; l'utente deve poter scegliere il colore dello sfondo e del testo al fine di ottenere la combinazione che gli permette la migliore visualizzazione possibile;
- Trasduzione di contenuti : nell'ottica della multimodalità, occorre implementare un'interfaccia vocale in grado di effettuare :
    - Text to speech : da testo a discorso, per proporre all'utente i contenuti testuali in modalità vocale (es. lettura di una voce di menu);
    - Speech to text : da discorso a testo, per riconoscere l'input vocale dell'utente (es. inserimento di userid e password in modalità vocale);
- Strategie di navigazione : per facilitare l'accesso dell'utente, occorre attuare delle particolari strategie di navigazione nell'applicazione :
    - Scansione sequenziale : possibilità di offrire all'utente una presentazione sequenziale dei contenuti (es. lettura sequenziale di una lezione);
    - Interfacciamento vocale : interfaccia vocale in grado di guidare l'utente nell'interazione con l'applicazione (es. navigazione attraverso comandi vocali, esplorazione della pagina guidata da output vocali);
- Semplificazione : l'interfaccia dell'applicazione deve essere semplificata nei seguenti aspetti :

- Semplificazione dell'interazione col sistema : l'interazione col sistema deve essere semplificata dall'accesso multimodale;
- Semplificazione dei contenuti : i contenuti dell'applicazione devono essere ridotti al minimo indispensabile per garntire le attività di consultazione e fruizione dei corsi;
- Semplificazione della navigazione e dell'accesso ai contenuti : la navigazione e l' accesso ai contenuti devono essere semplificati adottando particolari tecniche di interfacciamento quali : comandi vocali, esplorazione guidata da output vocali, lettura di contenuti didattici.

I principali requisiti funzionali che l'applicazione MultiLez i.NET deve soddisfare sono:

- Accesso alla piattaforma
- Consultazione corsi
- Fruizione corsi

Tenendo presente i requisiti generali citati sopra, l'interfaccia utente deve rispettare i seguenti criteri :

- Input e output : deve essere possibile un input/output sia in modalità vocale (speech) che attraverso l'utilizzo di periferiche hardware quali mouse e tastiera.
- Elementi dell'interfaccia sensibili : alcuni elementi dell'interfaccia ritenuti fondamentali, devo essere resi sensibili a determinati eventi (es. passaggio del mouse), in modo da poter dare un riscontro vocale; occorre porre attenzione però a non rendere eccessivamente verbosa l'applicazione.
- Contrasto : il contrasto ricopre un ruolo fondamentale; infatti un elevato contrasto di colori permette una migliore visualizzazione da parte dell'utente.
- Dimensione carattere : il carattere utilizzato deve avere dimensioni appropriate in grado di permettere una migliore visualizzazione da parte dell'utente;
- Informazioni : le informazioni presenti devono essere limitate ed avere una rappresentazione visuale con dimensioni adeguate e contrasto adeguato.
- Configurabilità contrasto/dimensione carattere : il contrasto che la dimensione del carattere devono essere resi configurabili da parte dell'utente; infatti si ritiene che i due criteri siano a discrezione dell'utente; un utente può avere una ricezione visiva migliore con un contrasto piuttosto che con un altro, oppure con un carattere più grande.

4.1.1 Accesso alla piattaforma

Durante la fase di accesso alla piattaforma, l'utente ha a disposizione le funzioni descritte nel diagramma seguente :

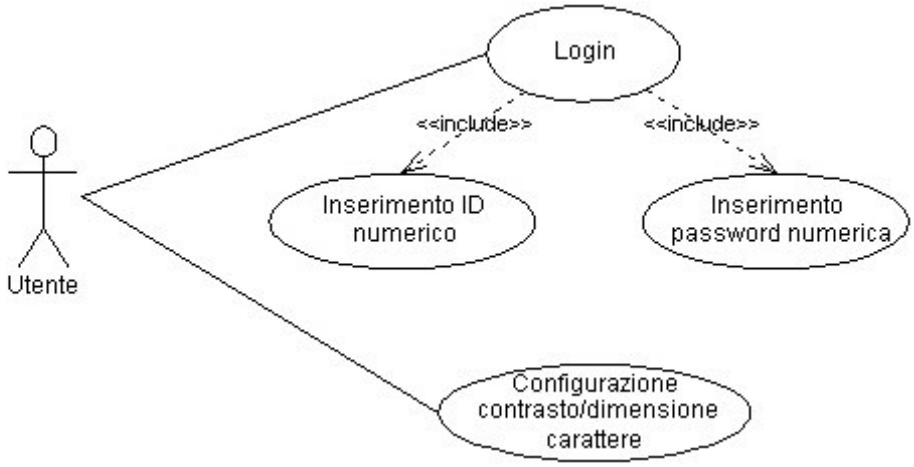

Figura 14 : Diagramma dei casi d'uso, accesso alla piattaforma

L'utilizzo di ID e password numerici piuttosto che alfanumerici dovrebbe facilitare il riconoscimento in caso di input vocale. La possibilità di configurare il contrasto e la dimensione del carattere deve essere disponibile anche nell'interfaccia di accesso al sistema.

CASO D'USO	**Login**
Descrizione :	L'utente si autentica con id numerico e password numerica ed entra nell'applicazione
Attore primario :	Utente
Precondizioni :	
Associazione con altri casi d'uso :	Include Inserimento id numerico ed Inserimento password numerica
Scenario principale :	1. L'utente trova il campo id numerico guidato dai messaggi vocali ed inserisce l'id numerico 2. L'utente trova il campo password numerica

	guidato dai messaggi vocali ed inserisce la password 3. L'utente trova il tasto login guidato dai messaggi vocali, clicca ed entra nell'applicazione
Scenario alternativo :	1. L'utente trova il campo id numerico guidato dai messaggi vocali ed inserisce l'id numerico 2. L'utente trova il campo password numerica guidato dai messaggi vocali ed inserisce la password 3. L'utente esegue il login con un comando vocale
Scenari errore :	• L'input vocale dell'utente non viene riconosciuto; l'operazione deve essere ripetuta • Id o password non sono corretti

CASO D'USO	**Configurazione contrasto/dimensione carattere**
Descrizione :	Possibilità di selezionare temi di contrasto predefiniti, e la dimensione carattere preferita
Attore primario :	Utente
Precondizioni :	
Associazione con altri casi d'uso :	
Scenario principale :	1. L'utente trova il tasto Configurazione, guidato da messaggi vocali, e clicca. 2. Navigazione guidata da messagi vocali della pagina di configurazione. 3. L'utente seleziona il tema interessato e/o la dimensione del carattere.

	4. Il tema di contrasto e/o la dimensione del carattere sono applicati, in modo che l'utente possa vedere il risultato.
Scenario alternativo :	
Scenari errore :	

CASO D'USO	**Inserimento ID numerico**
Descrizione :	Inserimento dell'ID numerico
Attore primario :	Utente
Precondizioni :	
Associazione con altri casi d'uso :	E' incluso in Login
Scenario principale :	1. L'utente è invitato da un messaggi o vocale, a dettare l'ID numerico. 2. Il testo riconosciuto è messo nel textbox relativo all'ID.
Scenario alternativo :	1. L'utente digita l'ID
Scenari errore :	L'input vocale dell'utente non è riconosciuto.

CASO D'USO	**Inserimento password numerica**
Descrizione :	Inserimento della password numerica
Attore primario :	Utente
Precondizioni :	
Associazione con altri casi d'uso :	E' incluso in Login
Scenario principale :	1. L'utente è invitato da un messaggio vocale, a dettare la password numerica. 2. Il testo riconosciuto è messo nel textbox relativo alla password.
Scenario alternativo :	L'utente digita la password
Scenari errore :	L'input vocale dell'utente non è riconosciuto.

4.1.2 Consultazione corsi

Le operazioni che l'utente può svolgere, una volta ottenuto l'accesso, sono illustrate nel diagramma dei casi d'uso seguente :

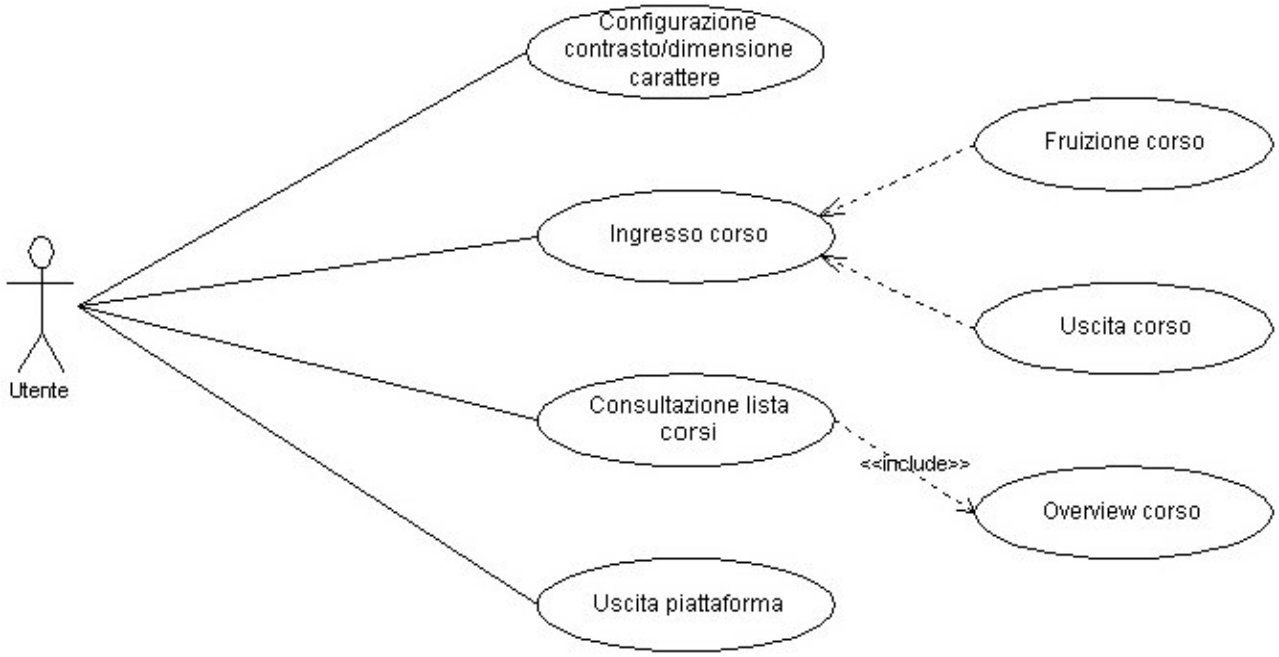

Figura 15 : Diagramma dei casi d'uso, consultazione corsi

CASO D'USO	**Configurazione contrasto/dimensione carattere**
Descrizione :	Possibilità di selezionare temi di contrasto predefiniti, e la dimensione carattere preferita
Attore primario :	Utente
Precondizioni :	L'utente deve aver effettuato il login
Associazione con altri casi d'uso :	
Scenario principale :	1. L'utente trova il tasto Configurazione, guidato da messaggi vocali, e clicca. 2. Navigazione guidata da messagi vocali della pagina di configurazione. 3. L'utente seleziona il tema interessato e/o la

	dimensione del carattere. 4. Il tema di contrasto e/o la dimensione del carattere sono applicati, in modo che l'utente possa vedere il risultato.
Scenario alternativo :	
Scenari errore :	

CASO D'USO	**Consultazione lista corsi**
Descrizione :	Possibilità di consultare la lista dei corsi disponibili
Attore primario :	Utente
Precondizioni :	L'utente deve aver effettuato il login
Associazione con altri casi d'uso :	Include overview corso
Scenario principale :	1. L'utente naviga con il mouse sulla lista corsi 2. Ogni elemento della lista dei corsi, ha un messaggio vocale associato relativo al nome del corso 3. L'utente clicca sul corso e viene presentata un'overview del corso
Scenario alternativo :	
Scenari errore :	

CASO D'USO	**Overview corso**
Descrizione :	Possibilità di consultare gli argomenti del corso
Attore primario :	Utente
Precondizioni :	L'utente deve aver effettuato il login
Associazione con altri casi d'uso :	E' incluso in overview corso
Scenario principale :	1. L'utente naviga con il mouse sulla lista degli argomenti

	2. Ogni elemento della lista degli argomenti, ha un messaggio vocale associato relativo al nome dell'argomento
Scenario alternativo :	
Scenari errore :	

CASO D'USO	**Ingresso corso**
Descrizione :	Accesso al corso corrente
Attore primario :	Utente
Precondizioni :	L'utente deve aver effettuato il login
Associazione con altri casi d'uso :	
Scenario principale :	1. L'utente trova il tasto per l'accesso al corso corrente, guidato dai messaggi vocali 2. L'applicazione reperisce il corso, tenendo conto del suo stato di avanzamento, e visualizza la tabella dei contenuti ed il contenuto corrente
Scenario alternativo :	1. L'utente accede al corso con un comando vocale 2. L'applicazione reperisce il corso, tenendo conto del suo stato di avanzamento, e visualizza la tabella dei contenuti ed il contenuto corrente
Scenari errore :	

CASO D'USO	**Fruizione corso**
Descrizione :	Possibilità di fruire i contenuti del corso
Attore primario :	Utente
Precondizioni :	L'utente deve aver effettuato il login, e deve aver

	effettuato l'accesso ad un corso
Associazione con altri casi d'uso :	Dipende dall'ingresso corso
Scenario principale :	1. L'utente naviga nei contenuti del corso 2. L'utente apprende i contenuti
Scenario alternativo :	
Scenari errore :	

CASO D'USO	**Uscita corso**
Descrizione :	Uscita dal corso corrente
Attore primario :	Utente
Precondizioni :	L'utente deve aver effettuato il login, e deve aver effettuato l'accesso ad un corso
Associazione con altri casi d'uso :	Dipende dall'ingresso corso
Scenario principale :	1. L'utente trova il tasto per la chiusura del corso, guidato dai messaggi vocali 2. L'applicazione aggiorna lo stato di avanzamento del corso e chiude la visualizzazione del corso
Scenario alternativo :	1. L'utente effettua la chiusura con un comando vocale 2. L'applicazione aggiorna lo stato di avanzamento del corso e chiude la visualizzazione del corso
Scenari errore :	

CASO D'USO	**Uscita piattaforma**
Descrizione :	Uscita dalla piattaforma
Attore primario :	Utente
Precondizioni :	L'utente deve aver effettuato il login

Associazione con altri casi d'uso :	
Scenario principale :	1. L'utente trova il tasto per l'uscita dalla piattaforma, guidato dai messaggi vocali 2. L'applicazione presenta la pagina di login
Scenario alternativo :	1. L'utente effettua l'uscita dalla piattaforma con un comando vocale 2. L'applicazione presenta la pagina di login
Scenari errore :	

4.1.3 Fruizione corso

La fruizione del corso prevede principalmente le attività descritte nel seguente diagramma :

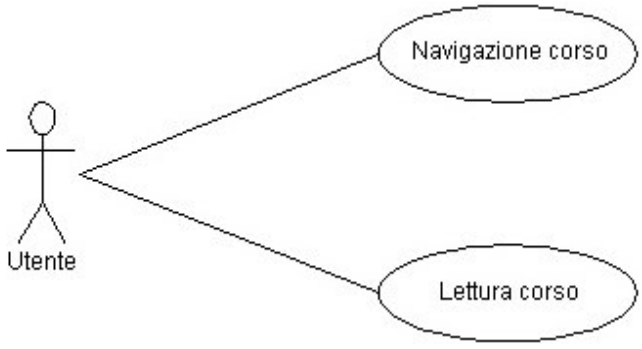

Figura 16 : Diagramma dei casi d'uso, fruizione corso

CASO D'USO	**Navigazione corso**
Descrizione :	Possibilità di consultare la lista dei contenuti del corso
Attore primario :	Utente
Precondizioni :	L'utente deve aver effettuato il login e aver effettuato l'accesso ad un corso
Associazione con altri casi d'uso :	Dipende dall'ingresso corso
Scenario principale :	1. L'utente naviga con il mouse sulla lista dei

	contenuti del corso 2. Ogni elemento della lista dei contenuti, ha un messaggio vocale associato relativo al nome del contenuto; in caso di nodo padre il messaggio specifica la possibilità di espandere il nodo 3. L'utente clicca sul contenuto e l'applicazione visualizza il contenuto
Scenario alternativo :	L'utente utilizza appositi comandi vocali per passare da un contenuto al successivo e viceversa
Scenari errore :	

CASO D'USO	**Lettura corso**
Descrizione :	Possibilità di apprendere i contenuti del corso
Attore primario :	Utente
Precondizioni :	L'utente deve aver effettuato il login, e deve aver effettuato l'accesso ad un corso
Associazione con altri casi d'uso :	Dipende dall'ingresso corso
Scenario principale :	1. L'utente trova il tasto per l'avvio della lettura contenuto del corso, guidato dai messaggi vocali 2. L'avanzamento della lettura è evidenziato da un differente contrasto testo/sfondo rispetto a quello dell'applicazione 3. L'utente può decidere di mettere in pausa la lettura per poi continuare in seguito 4. L'utente trova il tasto per lo stop della lettura contenuti del corso, guidato dai messaggi vocali

Scenario alternativo :	1. L'utente esegue l'avvio della lettura del contenuto di un corso mediante un comando vocale 2. L'avanzamento della lettura è evidenziato da un differente contrasto testo/sfondo rispetto a quello dell'applicazione 3. L'utente può decidere di mettere in pausa la lettura con un comando vocale per poi continuare in seguito sempre con un comando vocale 4. L'utente esegue lo stop della lettura con un comando vocale
Scenari errore :	

4.2 Progettazione di MultiLezi.NET

4.2.1 Integrazione con Lezi.NET

Alla luce dell'architettura di Lezi.NET vista nel paragrafo 3.1.2, l'applicazione MultiLe zi.NET ha il compito di implementare un front-tier in grado di offrire multimodalità . Quindi l'intervento sulla piattaforma Lezi.NET è limitato al front-tier, in quanto l'accesso ai contenuti deve avvenire attraverso lo stesso middle-tier e back-tier per garantire la sincronizzazione tra gli accessi.
Lo scenario che si andrebbe a formare è visibile nella figura seguente :

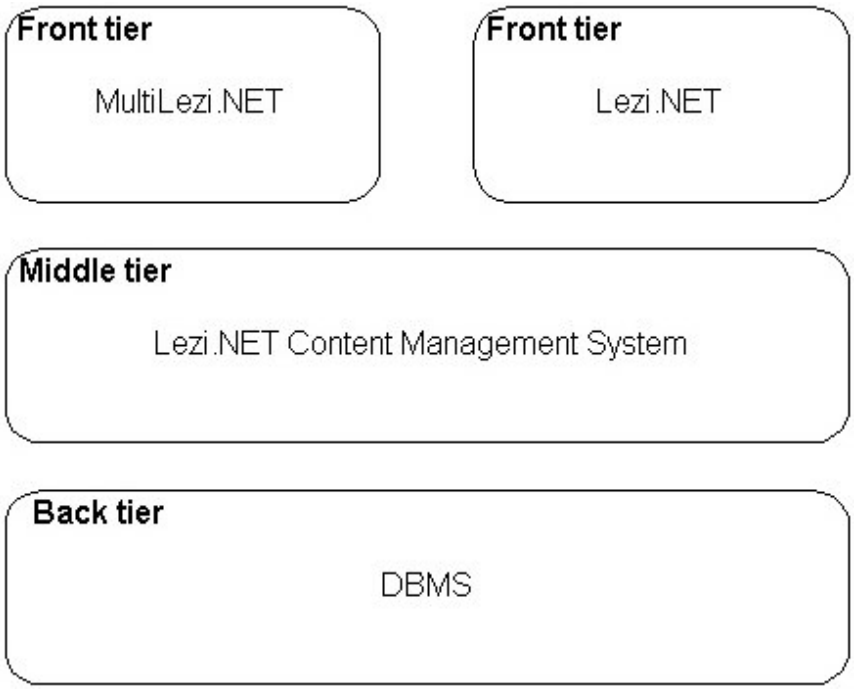

Figura 17 : Integrazione tra Lezi.NET e MultiLezi.NET

Tra le varie funzionalità di Lezi.NET, quella implementata da MultiLezi.Net è la fruizione dei corsi. Infatti lo scopo di MultiLezi.NET è di mettere a disposizione un'interfaccia multimodale per la fruizione dei corsi. Per quanto riguarda le altre funzioni di amministrazione di sistema e gestione di corsi e utenti è sempre possibile utilizzare Lezi.NET.

La scelta di utilizzare per lo sviluppo di Lezi.NET tecnologie Microsoft come ASP.NET e C#, ha influenzato le scelte per MultiLezi.NET sia in ambito di tecnologia per lo speech che per la tecnologia web.

Infatti come visto nel capitolo 3, la Microsoft mette a disposizione uno Speech Application SDK per lo sviluppo di interfacce vocali e multimodali, che prevede l'integrazione con ASP.NET e C#. Inoltre il proxy per l'accesso ai web services del CMS è scritto in C#.

Quindi MultiLezi.NET è sostanzialmente un'applicazione web basata su tecnologia ASP.NET, C# e JavaScript, in grado di generare pagine HTML+SALT ed implementare il run-time SCORM per il tracking dei corsi.

L'architettura dettagliata di MultiLezi.NET è evidenziata nella figura seguente :

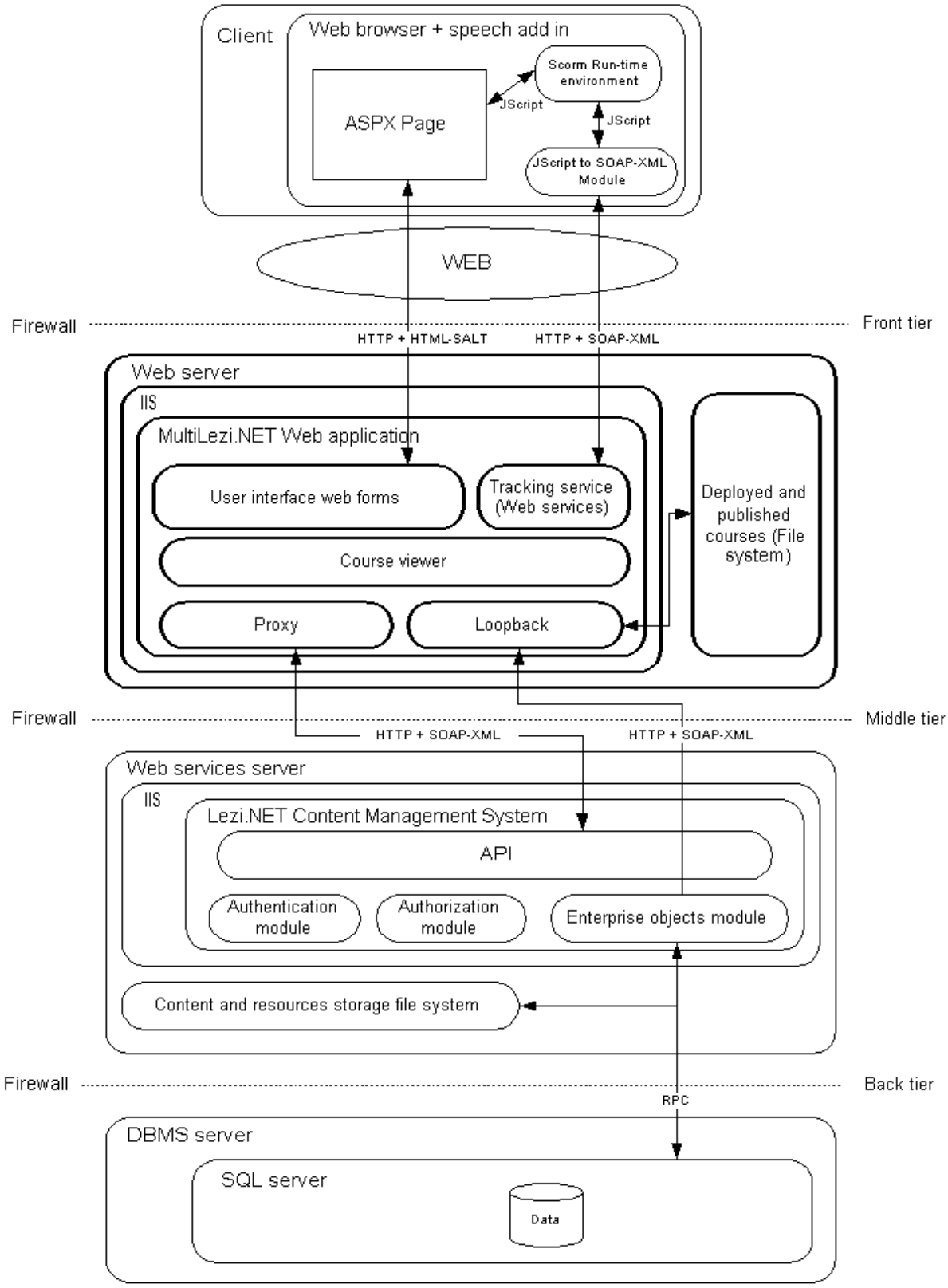

Figura 18 : Architettura MultiLezi.NET

4.2.2 Struttura dell'applicazione web

MultiLezi.NET è una applicazione web, basata sulla tecnologia .NET, per la fruizione di corsi in un ambiente web-based con accesso multimodale. Si avvale delle specifiche SCORM per il tracciamento dei percorsi di apprendimento, e della tecnologia SALT per l'implementazione dell'interfaccia vocale integrata a quella grafica.
In particolare si tratta di pagine web ASP.NET per l'implementazione dell'interfaccia grafica e vocale, un run-time SCORM per il tracciamento dei percorsi di apprendimento implementato in JavaScript e Web Services, ed un proxy per la comunicazione con i Web Services esposti dal CMS Lezi.NET.
Le pagine chiave dell'applicazione sono le seguenti :

- Login : pagina per l'autenticazione dell'utente; permette l'inserimento di userid e password per l'autenticazione nel sistema;
- Homepage : home page dell'applicazione, in cui è possibile consultare l'elenco completo dei corsi, ed eventualmente sceglierne uno;
- Viewer : pagina che visualizza il corso selezionato nella homepage; permette la navigazione degli argomenti del corso.
- Config : pagina per la configurazione del contrasto sfondo/carattere, e della dimensione del carattere

La figura seguente presenta una mappa dell'applicazione web.

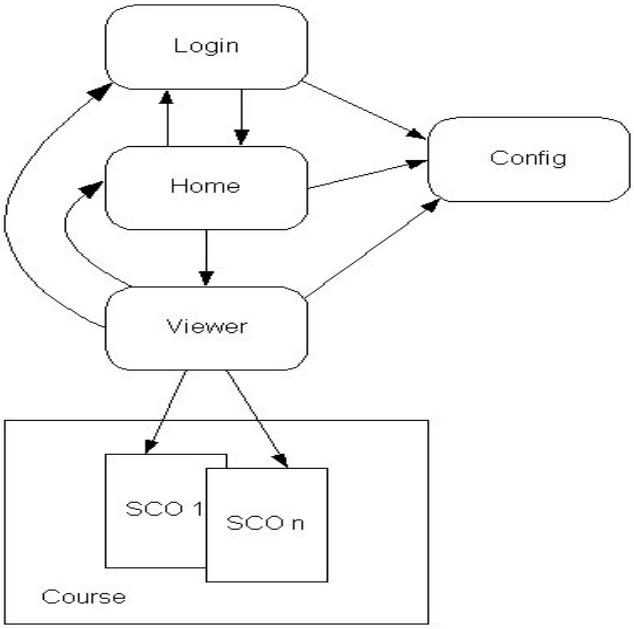

Figura 19 : Mappa applicazione web MultiLezi.NET

Le frecce in figura 19 rappresentano la navigabilità dell'applicazione. Dalla pagina di Login è possibile accedere alla pagina Config per le opzioni di configurazione. Fornendo UserID e Password validi si accede invece alla Homepage, in cui è presente l'elenco dei corsi e la descrizione dei relativi argomenti. Le impostazioni di configurazione sono mantenute nel passaggio da una pagina all'altra, e anche dalla Homepage è possibile accedere alla pagina Config. Lanciando un determinato corso si accede alla pagina Viewer che permette la visualizzaione e la navigazione degli argomenti del corso. Per ogni argomento la pagina Viewer si occupa di caricare uno SCO, ovvero una pagina HTML+SALT. Dalla pagina Viewer è possibile tornare alla Homepage, accedere alla pagina Config oppure effettuare il logout dall'applicazione. Naturalmente anche dalla Homepage è possibile effettuare il logout.

5 IMPLEMENTAZIONE DELL'APPLICAZIONE MULTILEZI.NET

5.1 Autenticazione

Per accedere all'applicazione MultiLezi.NET, l'utente deve eseguire l'autenticazione fornendo un identificativo utente (userID) ed una password.
Per implementare il meccanismo di autenticazione è stato utilizzato il metodo di autenticazione "Form" messo a disposizione da .NET Framework per ASP.NET. Si tratta di un sistema sicuro e collaudato per la fase di autenticazione che consiste sostanzialmente nel reindirizzamento verso una pagina per effettuare il login nel caso in cui arrivi una richiesta ad una pagina dell'applicazione da un utente non autenticato. La pagina per effettuare il login è Login.aspx mentre la pagina principale per l'avvio dell'applicazione è Default.aspx. Un utente non autenticato che tenterà l'accesso a Default.aspx o alle altre pagine dell'applicazione sarà reindirizzato alla pagina Login.aspx.
I passi da effettuare per l'autenticazione sono i seguenti :

- Richiesta della pagina Default.aspx;
- Reindirizzamento effettuato dal sistema di autenticazione "Forms" verso la pagina Login.aspx;
- Inserimento userID e password;
- Verifica delle credenziali fornite tramite il CMS Lezi.NET;
- Reindirizzamento verso la pagina Default.aspx nel caso di credenziali corrette;
- Invio al client dell'utente di un token di sessione (hash calcolato in base al nome utente, userID, password, data e ora di sistema) che sarà memorizzato in un cookie e riutilizzato in seguito per la richiesta delle varie pagine dell'applicazione.

La pagina Login.aspx è composta principalmente da web components per la grafica come textbox, button, link, e speech components (Speech Application SDK) per l'interfaccia vocale come prompt e listen. La figura seguente mostra uno schema della pagina con i relativi componenti.

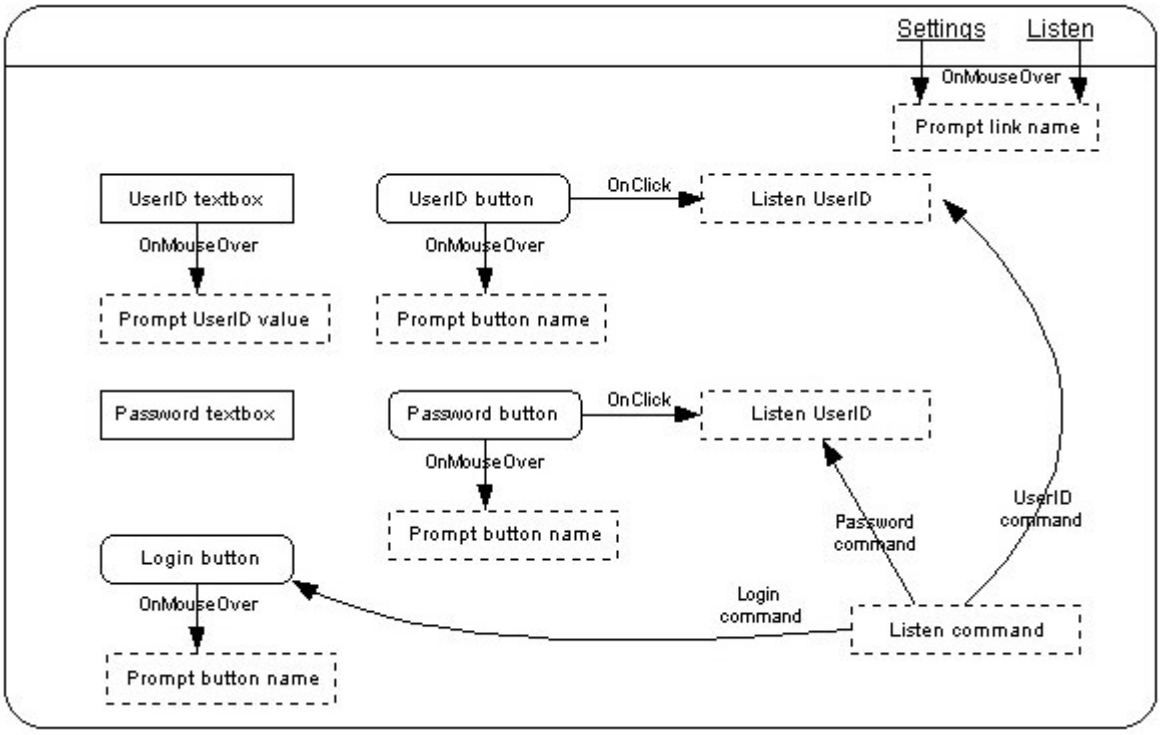

Figura 20 : Schema della pagina Login.aspx

I texbox sono adibiti all'inserimento di userID e password, mentre due button permettono di attivare il riconoscimento vocale rispettivamente di userID e password. Un altro button permette di effettuare il login eseguendo il post della pagina. Esiste inoltre un link alla pagina di configurazione ed un link per l'attivazione del listen dei comandi di cui sarà spiegata la funzione in seguito.

5.1.1 Navigazione guidata da output vocali

Per ogni componente grafico della pagina esiste un componente prompt che è attivato dall'evento OnMouseOver. Ogni prompt è impostato con la descrizione del componente a cui si riferisce, e al verificarsi dell'evento OnMouse Over viene generato un output vocale relativo alla descrizione. In questo modo la navigazione della pagina può essere guidata da ouput vocali che identificano il componente che si sta puntando.
Il codice seguente mostra la definizione di un prompt relativo al taso userID nella pagina Login.aspx :

```
<speech:prompt id="PromptUserID" runat="server" Width="198px">
          <InlineContent>User ID</InlineContent>
</speech:prompt>
```

I prompt vengono processati lato server alla chiamata della pagina per la generazione dei tag SALT, mentre la loro esecuzione è gestita lato client tramite codice JavaScript; lanciando il metodo Start() dell'oggetto JavaScript associato al componente prompt, è possibile eseguire l'output vocale.
La definizione del button con l'evento OnMouseO ver è la seguente :

```
<INPUT onmouseover="PromptUserID.Start()" type="button" value="UserID">
```

5.1.2 Riconoscimento vocale delle credenziali utente

Per il riconoscimento vocale di userID e password sono utilizzati due speech components listen. Il riconoscimento è attivabile tramite la pressione del button corrispondente oppure tramite comando vocale come vedremo nel paragrafo seguente.
L'esecuzione del componente listen è gestibile lato client utilizzando codice JavaScript; come per i prompt, esiste un metodo Start() dell'oggetto JavaScript associato al componente. L'eventuale testo riconosciuto, può essere collegato a componenti della pagina come ad esempio textbox, mediante l'impostazione della proprietà bindings del componente listen. Al componente list en dello userID è collegato il texbox dello userID, mentre al listen della password è collegato il textbox della password. Il risultato che si ottiene è che dettando lo userID, viene visualizzato nel texbox userID.
Il codice seguente permette la definizione del listen relativo allo userID :

```
 <speech:listen id="ListenUserID" runat="server" Width="141px"
InitialTimeout="2500"
    EndSilence="2500" MaxTimeout="5000" BabbleTimeout="2500">
         <Grammars>
             <speech:Grammar Src="Grammars/Library.grxml#Alphanum3"
                     ID="ListenUserID_Grammar1">
             </speech:Grammar>
        </Grammars>
       <Bindings>
    <speech:Bind TargetAttribute="value" Value="/SML"
              TargetElement="userIDTextBox">
           </speech:Bind>
      </Bindings>
  </speech:listen>
```

Per il riconoscimento occorre comunque definire una grammatica che definisca i possibili input dell'utente. In questo caso è stata utilizzata una grammatica predefinita che permette di riconoscere sequenze alfanumeriche di lunghezza massima 3.

5.1.3 Comandi vocali per l'autenticazione

Per il riconoscimento di comandi vocali in fase di autenticazione, è stato utilizzato un componente listen con una grammatica personalizzata.
Infatti i comandi disponibili sono sostanzialmente i seguenti :

- Login : per l'esecuzione del post della pagina; equivale alla pressione del button Login;
- UserID : per l'esecuzione del riconoscimento vocale userID; equivale alla pressione del button UserID;
- Password : per l'esecuzione del ri conoscimento vocale password; equivale alla pressione del button Password.

Il componente listen è mandato in esecuzione al caricamento della pagina permettendo così il riconoscimento dell'input vocale in qualsiasi istante. Per fare ciò il componente listen deve essere impostato in modalità multipla. A volte il processo listen si interrompe per cause ignote; per questo è stato predisposto un link per rieseguire lo Start del componente listen, come già accennato nel primo paragrafo.
Associati al componente listen ci sono degli eventi al verificarsi dei quali è possibile eseguire delle azioni. Al verificarsi dell'evento OnClientReco, è possibile eseguire del codice JavaScript; in particolare si possono verificare due scenari a seconda del comando riconosciuto :

- UserID/Password : deve essere interrotto il riconoscimento del componente listen relativo ai comandi, e deve essere attivato il riconoscimento del componente listen relativo allo userID o password; al termine del riconoscimento deve essere riattivato il listen dei comandi al fine di permettere il riconoscimento di ulteriori comandi;
- Login : deve essere effettuato il post della pagina.

```
<speech:listen id="ListenCommand" runat="server" Width="145px"
    OnClientReco="commandReco()"  Mode="Multiple">
 <Grammars>
          <speech:Grammar Src="Grammars/MultiLeziNET.grxml#Command"
              ID="ListenCommand_Grammar1">
```

```
          </speech:Grammar>
</Grammars>
</speech:listen>
```

Le funzioni JavaScript utilizzate per la gestione del riconoscimento sono le seguenti :

```
function commandReco(){
    switch(ListenCommand.text) {
          case "UserID":
                ListenUI();
              break;
          case "Password":
                ListenPW();
                  break;
          case "Login":
            EnterApplication();
                  break;
          default:
                  break;
    }
}

function EnterApplication(){
    ListenCommand.Stop();
      Form1.submit();
}

function ListenUI(){
     ListenCommand.Stop();
     PromptInsertUI.Start();
     ListenUserID.Start();
}

function ListenPW(){
     ListenCommand.Stop();
     PromptInsertPW.Start();
     ListenPassword.Start();
}
```

La grammatica da associare al componente listen deve permettere il riconoscimento delle tre parole : login, userID, password. In pratica devono essere definite 3 regole grammaticali, una per ciascuna parola. Il codice relativo alla grammatica personalizzata è il seguente :

```
<rule id="Login" scope="private">
      <item>Login</item>
      <tag>$._value = "Login"</tag>
</rule>
<rule id="Password" scope="private">
      <item>Password</item>
      <tag>$._value = "Password"</tag>
```

```
</rule>
<rule id="UserID" scope="private">
      <item>UserID</item>
      <tag>$._value = "UserID"</tag>
</rule>
<rule id="Command" scope="public">
      <tag>$.Command = {}</tag>
      <one-of>
            <item>
                  <ruleref uri="#Password" type="application/srgs+xml"/>
                  <tag>$.Command.Password = $$</tag>
            </item>
            <item>
                  <ruleref uri="#UserID" type="application/srgs+xml"/>
                  <tag>$.Command.UserID = $$</tag>
            </item>
            <item>
                  <ruleref uri="#Login" type="application/srgs+xml"/>
                  <tag>$.Command.Login = $$</tag>
            </item>
      </one-of>
</rule>
```

La regola Command raggruppa le tre regole relative ai comandi e permette una selezione esclusiva.

Di seguito è riportato il digramma di sequenza dell'autenticazione attraverso comandi vocali.

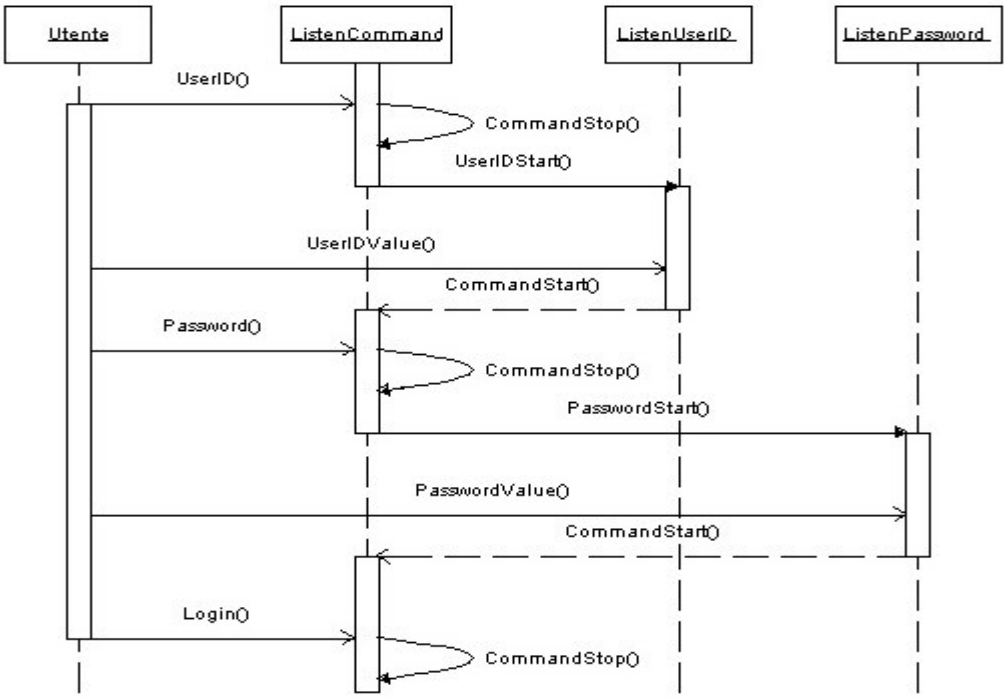

Figura 21: Diagramma di sequenza, autenticazione attraverso comandi vocali

5.2 Homepage dell'applicazione

Una volta effettuata l'autenticazione si ottiene l'accesso all'applicazione, che si presenta con una Homepage in cui è possibile consultare la lista dei corsi.
La pagina principale dell'applicazione è Default.aspx; si tratta di una pagina che definisce un frameset composto da un top frame comune a tutte le pagine dell'applicazione, e da un center frame che ospita tutte le varie pagine.

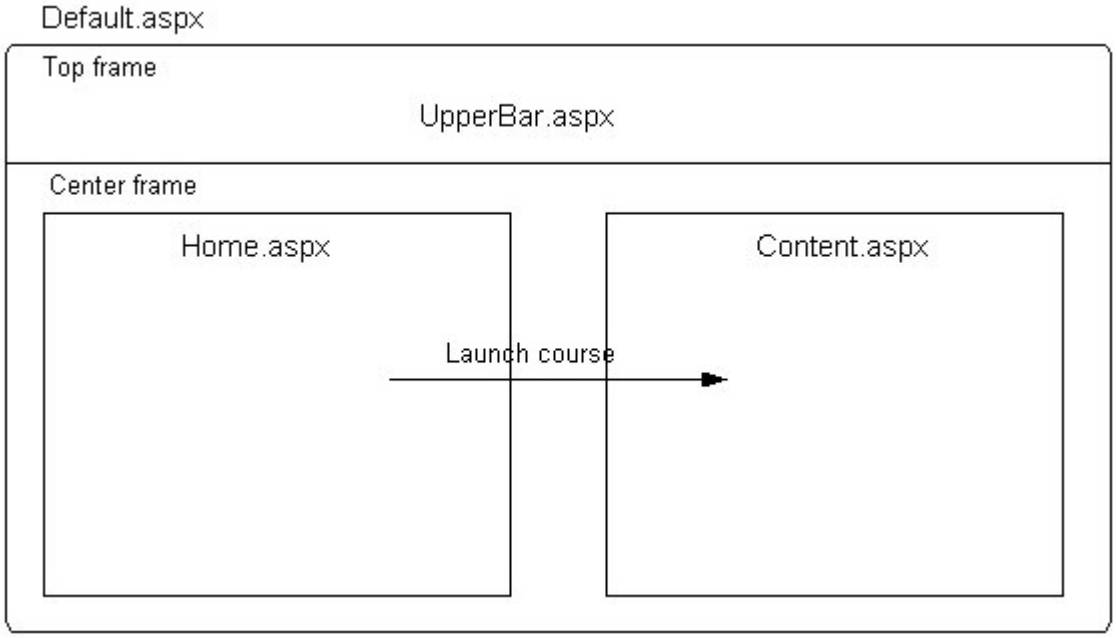

Figura 22 : Schema della pagina Default.aspx

Il top frame contiene la pagina UpperBar.aspx che definisce i componenti per i comandi vocali e visualizza dei link di gestione applicazione. In particolare sono definiti un comando listen in modalità multipla, una grammatica personalizzata e la logica in JavaScript per la gestione dei comandi vocali. I link sono per il reindirizzamento alla Homepage, per la pagina di configurazione e per il logout.
La pagina caricata all'avvio è Home.aspx, e permette la consultazione dei corsi. È anch'essa composta da due frame, uno di sinistra con l'elenco dei corsi ed uno di destra con la descrizione degli argomenti del corso selezionato. Nel caso in cui non ci sia un corso selezionato, il frame di destra visualizza una pagina che presenta un riferimento all'ultimo corso lanciato (Start.aspx).
Lanciando un corso, nel frame centrale viene caricata la pagina Content.aspx che si occupa di caricare tutti i componenti per la visualizzazione dei contenuti del corso.

5.2.1 Menu principale dell'applicazione

Il menu principale dell'applicazione è implementato dalla pagina UpperBar.aspx che è caricata nel top frame della pagina principale dell'applicazio ne Default.aspx.
Lo schema della pagina è il seguente :

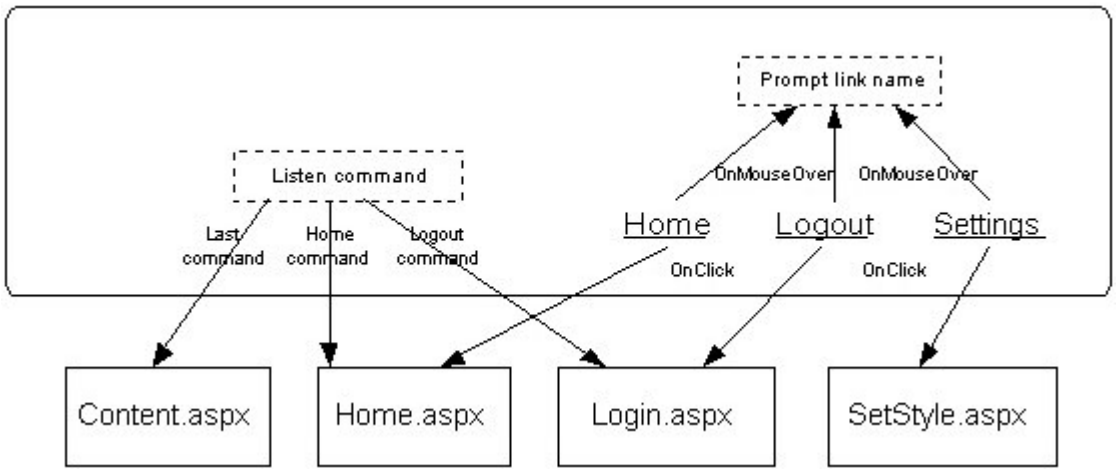

Figura 23 : Schema UpperBar.aspx

Come già accennato nell'introduzione del capitolo sono presenti i seguenti link :

- Home : carica la pagina Home.aspx nel center frame, permettendo di visualizzare l'elenco dei corsi;
- Logout : carica la pagina Logout.aspx che effettua le operazioni per la chiusura della sessione e la cancellazione del token per l'accesso al CMS, ed esegue la redirezione verso la pagina Login.aspx per un eventuale nuovo accesso;
- Settings : carica la pagina SetStyles.aspx per la configurazione di contrasto e dimensione carattere.

I link possono essere riconosciuti grazie a dei componenti prompt che producono output vocali con la relativa descrizione al verificarsi dell'evento OnMouseOver.
Inoltre è presente un componente listen che permette il riconoscimento di alcuni comandi vocali per l'esecuzione di funzioni dell'applicazione.
L'ascolto è fatto partire al caricamento della pagina, ed essendo in modalità multip la, permette il riconoscimento di più comandi consecutivi. Naturalmente come per la fase di autenticazione è stato necessario definire una grammatica personalizzata per il riconoscimento dei comandi.

I comandi definiti sono i seguenti :

- Homepage : per la visualizzazione della pagina Home.aspx (equivale al click del link Home);
- Logout : per il logout dall'applicazione (equivale al click del link Logout);
- Launch : per il lancio dell'ultimo corso lanciato;
- Comandi abilitati solo in fase di fruizione corso :
    - Back : contenuto precedente;
    - Next : contenuto successivo;
    - Close : chiusura corso;
    - Start : inizio lettura contenuto;
    - Stop : fine lettura contenuto;
    - Wait : sospensione lettura contenuto;
    - Continue : proseguimento lettura contenuto.

Come si può vedere alcuni comandi sono globali (validi in ogni fase) mentre alcuni sono specifici per la fase di fruizione del corso.
Le tecniche di sviluppo dei componenti sono simili a quelle utilizzate per l'autenticazione, e quindi consultabili nel paragrafo 5.1.

5.2.2 Consultazione corsi guidata da output vocale

La consultazione dei corsi è implementata dalla pagina Home.aspx, un frameset composto principalmente da una pagina con l'elenco dei corsi (BarMenu.aspx) ed una pagina con l'elenco degli argomenti del corso (Corse.aspx).
Lo schema della pagina è rappresentato in figura :

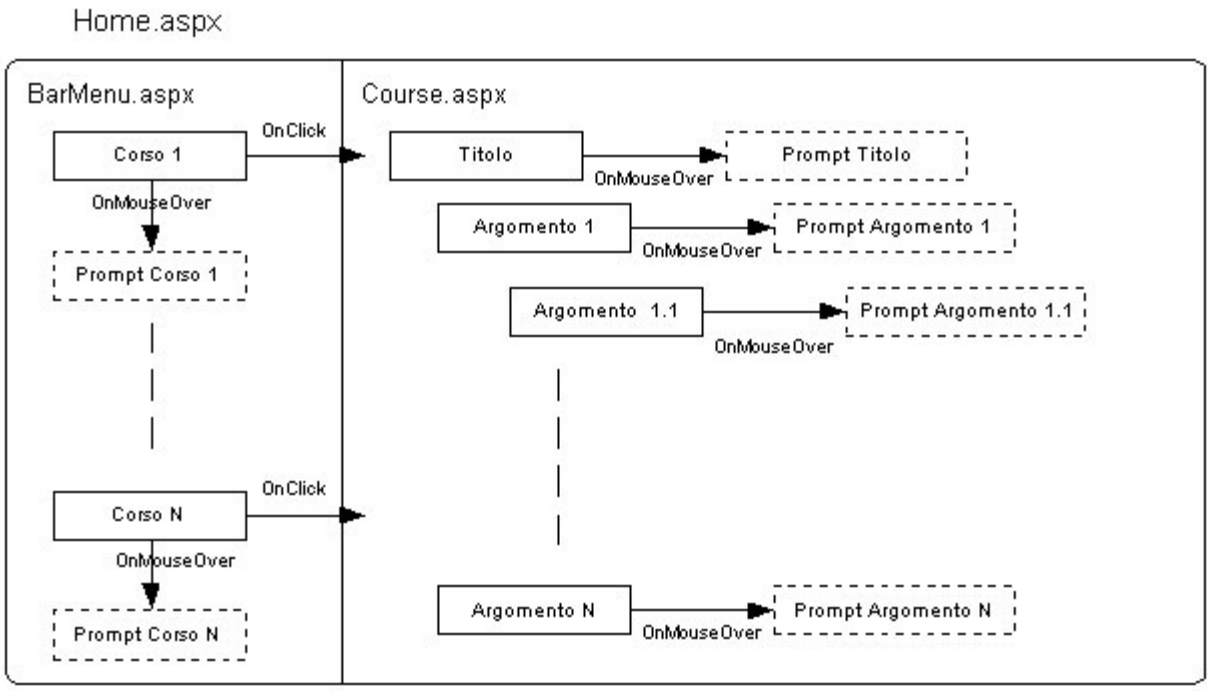

Figura 24 : Schema Home.aspx

BarMenu.aspx visualizza in un web component DataList l'elenco dei corsi ritornato dal web method GetCourseList() messo a disposizione dal CMS. Course.aspx invece legge l'elenco degli argomenti dall'IMSManifest.xml e lo visualizza in una tabella. La pagina Course.aspx viene ricaricata con gli argomenti aggiornati ogni volta che viene selezionato un corso dall'elenco dei corsi.

La navigazione dei corsi e dei relativi argomenti è guidata da output vocale, grazie alla definizione di componenti prompt relativi agli elementi di ciascun elenco. In questo caso però i vari corsi disponibili ed i relativi argomenti sono caricati nella fase di load della pagina rendendo impossibile una definizione statica dei componenti prompt per l'esecuzione dell'output vocale.

La definizione dei componenti prompt avviene quindi lato server utilizzando classi C# messe a disposizione da Speech Application SDK, nella fase di creazione degli elenchi.

Un esempio di codice utilizzato per la creazione è il seguente :

```
promptItem = new Microsoft.Speech.Web.UI.Prompt();
promptItem.InlineContent.Controls.Add(new LiteralControl(item.title));
promptItem.ID = "prompt"+item.identifier;
td.Controls.Add(new LiteralControl("<a id=\"item"+item.identifier+
 "\"onmouseover=\"promptItem('"+promptItem.ID+"')\">"+item.title+"</a>"));
td.Controls.Add(promptItem);
```

```
tr.Cells.Add(td);
tocTable.Rows.Add(tr);
```

L'esempio è relativo all'inserimento di un elemento (item) nell'elenco degli argomenti del corso. Il prompt viene creato utilizzando il metodo costruttore della classe Microsoft.Speech.Web.Prompt; in seguito vengono impostati il testo (InlineContent) e l'identificatore. Inoltre è aggiunto l'evento OnMouseOver nella definizione del tag html che visualizzerà l'argomento nella tabella. Al verificarsi dell'evento OnMouseOver viene chiamata una funzione JavaScript cui viene passato l'ID del prompt che deve essere mandato in esecuzione.

5.3 Visualizzatore dei corsi

L'ambiente di fruizione dei corsi è implementato dalla pagina Content.aspx che è caricata nel frame centrale della pagina Default.aspx, come già visto nel paragrafo precedente.
Per poter essere IMS-SCORM compatibile, l'ambiente deve fornire all'untente gli strumenti per interagire con i contenuti, ed il run-time per permettere ai contenuti di interagire con il tracking-service dell'LMS. E' dunque necessario creare due canali di comunicazione : un canale di distribuzione, su protocollo http, mediante il quale il browser richiede ed ottiene i contenuti da visualizzare; un canale di tracking, implementato tramite web services, che permette la comunicazione tra run-time client side e LMS.
La pagina Content.aspx è composta da diversi frame come si può vedere nella figura seguente :

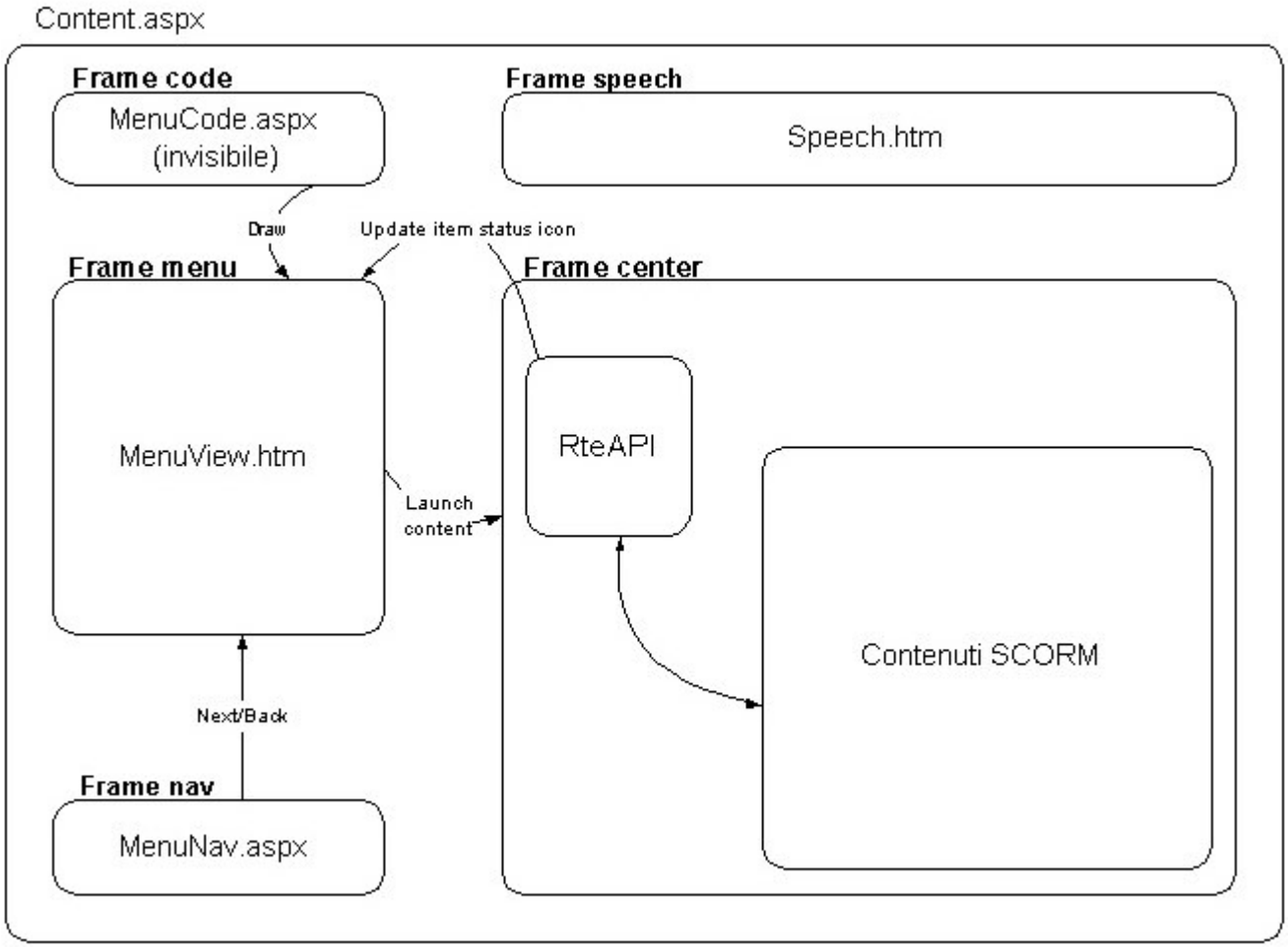

Figura 25: Schema della pagina Content.aspx

Il frame code ospita la pagina MenuCode.aspx che ha il compito di generare dinamicamente client-side un menu ad albero dei contenuti del corso (nel frame menu). Per fare questo sono utilizzate delle classi JavaScript che hanno il compito di mantenere lo stato degli item del menu e di ridisegnare il menu (riscrivendo la pagina MenuView.htm) ogni qual volta si ha un cambiamento di stato.

I contenuti dell'albero vengono inizialmente letti server-side dall'IMSmanifest.xml producendo una stringa contenete codice JavaScript da eseguire client-side durante l'inizializzazione della pagina. Si tratta di codice necessario ad inizializzare le classi JavaScript del menu. Successivamente, ad ogni cambiamento di stato, saranno chiamati i metodi per disegnare il menu sugli oggetti precedentemente instanziati.

Il frame centrale ospita la pagina relativa al contenuto SCORM in esecuzione; questa pagina utilizza l'istanza della classe RteAPI per la comunicazione con l'LMS. L'oggetto api è instanziato nella fase di

inizializzazione della pagina Content.aspx, in modo da essere disponibile ai vari contenuti che saranno caricati.

RteAPI è una classe JavaScript che implementa le API richieste dal run-time SCORM. Questa classe fornisce l'interfaccia di comunicazione tra i contenuti SCORM e l'LMS. Siccome il tracking service del LMS in MultiLeziNET è implementato mediante Web service è necessario che, una volta chiamato un metodo da parte de contenuto, sia costruito un pacchetto XML SOAP in modo da invocare il metodo corrispondente sul tracking service. Ad esempio, se è invocato il metodo LMSInitialize() utilizzando l'oggetto api.LMSIntialize(), il corpo di tale metodo costruirà un messaggio XML SOAP che invocherà il corrispondente Web service di nome LMSInitialize passando correttamente i parametri forniti dal contenuto. L'API per la creazione, invio e ricezione di messaggi XML SOAP è realizzata dalla classe WebServiceHTC (DHTML behaviors).

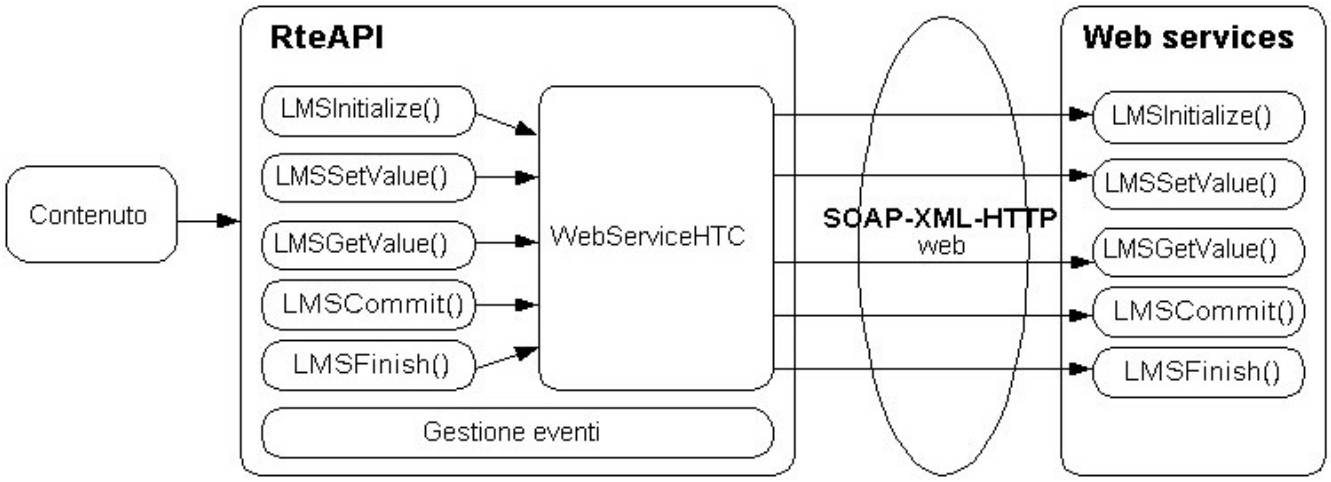

Figura 26: Schema di integrazione tra contenuto, api del run-time SCORM e web services LMS

Il frame nav ospita la pagina MenuNav.aspx che mette a disposizione i pulsanti per la navigazione dei contenuti (next,back) e per la chiusura del corso (redirezione alla pagina Home.aspx).

Il frame speech infine, contiene la pagina speech.htm che permette la gestione della lettura dei contenuti dei corsi. Si tratta sostanzialmente di 4 button che permettono:

- Start : avvio della lettura del contenuto corrente;
- Stop : arresto della lettura del contenuto corrente;
- Wait : pausa della lettura del contenuto corrente;
- Continue : proseguimento della lettura del contenuto corrente;

Nei paragrafi seguenti sarà spiegata l'implementazione dell'interfaccia vocale.

5.3.1 Guida vocale delle funzioni del visualizzatore

Anche per la pagina del visualizzatore sono stati utilizzati dei componenti prompt per la guida vocale dei principali comandi presenti nella pagina e per il menu ad albero dei contenuti.
Per quanto riguarda i frame nav e speech sono stati utilizzati dei speech components prompt come già era stato fatto per le altre pagine.
Invece per il menu la questione è leggermente complicata dal fatto che è disegnato a run-time lato client da classi JavaScript. La soluzione intrapresa è stata quella di aggiungere codice per la gestione dell'evento OnMouseOver, nel metodo (delle classi del menu) che si occupa di creare il tag HTML per il testo dei vari contenuti. In pratica non è creato un elemento prompt per ogni contenuto, bensì un solo elemento prompt nella pagina MenuCode.aspx, al quale viene passato il testo del contenuto per essere eseguito. Infatti oltre al metodo Start(), il componente prompt mette a disposizione il metodo Start(param), al quale può essere passato come parametro il testo da eseguire.
Di seguito si può vedere il codice del metodo makeText() della classe SbItem :

```
function SBItem.prototype.makeText()
{
   out += '<span id="'+this.s_path+'" title="' + this.s_text + '" class="sel"
onmouseover="parent.'+codeFrameId+'.MenuLessons.Start(&quot ' + this.s_text + '
&quot)" onclick="parent.'+codeFrameId+'.'+menuManI+'.select(&quot' + this.s_path
+ '&quot)">'+ this.s_text +'</span>\n';

}
```

La variabile out contiene tutto il codice HTML relativo al menu da visualizzare, e con questa istruzione, viene aggiunto il tag relativo ad un contenuto. In grassetto si può vedere il codice dell'evento OnMouseOver; la variabile codeFrameId contiene il nome del frame a cui fa r riferimento (code), mentre MenuLessons è il nome dell'elemento prompt della pagina MenuCode.aspx contenuta nel frame code; infine al metodo start viene passato come parametro il testo del contenuto.
Nella pagina MenuView.htm che verrà visualizzata, il contenuto sarà così definito :

```
<span id="_0" title="Maritime Navigation" class="sel"
onmouseover="parent.code.MenuLessons.Start(&quot Maritime Navigation &quot)"
onclick="parent.code.menuMan.select(&quot_0&quot)">Maritime Navigation</span>
```

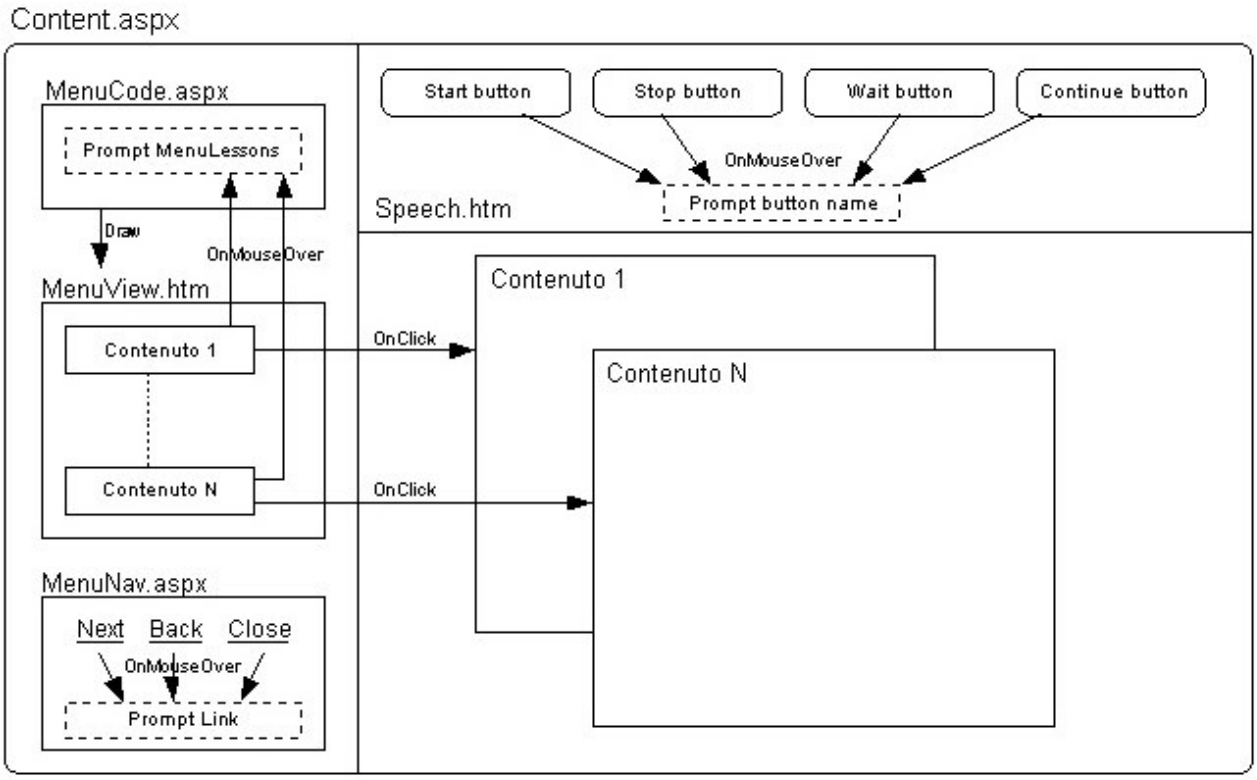

Figura 27: Mappa della pagina Content.aspx con speech components

5.3.2 Lettura dei contenuti

La funzionalità di lettura dei contenuti consiste nell'esecuzione di un output vocale relativo al testo del contenuto selezionato. Per lo sviluppo del lettore, è stato preso come riferimento il corso "Maritime Navigation", interamente composto da pagine HTML.

Le funzioni che il lettore implementa sono le seguenti :

- Start : inizio lettura di un contenuto;
- Stop : fine lettura di un contenuto;
- Wait : sospensione lettura di un contenuto;
- Continue : proseguimento lettura di un contenuto;
- Evidenziatore : segnalazione dello stato di avanzamento della lettura.

Trattandosi di pagine HTML, non è stato possibile utilizzare speech component prompt, utilizzabili solo in pagine ASP.NET. E' stato quindi necessario utilizzare tag <salt> integrati nelle pagine html. In particolare è stato utilizzato un unico tag <salt:prompt> al quale viene assegnato dinamicamente il testo da eseguire nell'output vocale. Oltre al metodo Start(param), l'oggetto prompt è anche dotato dei metodi Pause( ) (per la funzione wait), Resume( ) (per la funzione continue) e Stop( ) (per la funzione stop).

La tecnica utilizzata per ottenere il testo da leggere è stata quella di inserire nelle varie pagine dei tag <INPUT> di tipo nascosto. In pratica il testo totale della pagina è stato suddiviso in blocchi ed ogni blocco di testo è stato assegnato all'attributo value di un tag <INPUT>.

La suddivisione è stata effettuata per poter implementare la funzione di notifica dello stato di avanzamento della lettura. Infatti il blocco di testo corrente viene evidenziato invertendo i colori di testo e sfondo.

Per capire meglio vediamo un esempio :

```
<BODY>
    <?import namespace="salt" implementation="#Speechtags" />
    <form id="formspeech" name="formspeech">

            <salt:prompt id="promptText" oncomplete="nextText()">
            </salt:prompt>

    </form>
    <form id="formtext" name="formtext">

      <INPUT id="HiddenText" type="hidden" size="81" value="SCO 01.
The purpose of this course is to demonstrate the functionality and capability of
the ADL SAMPLE Run-time Environment. CTC does not recommend, propose or otherwise
promote the style, fashion, or type of content presented in this course."
NAME="HiddenText">

        <INPUT id="HiddenText" type="hidden" size="81" value="
Inland Rules of the Road. REFERENCE: U.S. Coast Guard, Commandant Instruction
M16672.2C . The material of this course is of the U.S Coast Guard's Rules of the
Road in compliance with U.S. Regulations." NAME="HiddenText">

        <INPUT id="HiddenText" type="hidden" size="81"  value="
LESSON OBJECTIVE: This course will give the student a basic understanding of the
Inland Rules of Navigation. These rules have been Coast Guard approved according to
the instruction listed above and U.S. Law.
Enjoy the course " NAME="HiddenText">

      </form>
       <FONT color="lightslategray"><B><I>
                    <LABEL>
  <a name="divisione">
       <P align="right">SCO 01</LABEL></I></B> </FONT></P>
       <P>
```

```
                <B><I>The purpose of this course is to demonstrate the
functionality and capability of
                        the ADL SAMPLE Run-time Environment. CTC does not
recommend, propose or
                        otherwise promote the style, fashion, or type of
content presented in this
                        course.</I></B>
            </P>
        </a>
        <a name="divisione">
            <H1>Inland Rules of the Road</H1>
            <HR>
            <P>
                <B>REFERENCE:</B> U.S. Coast Guard, Commandant Instruction
M16672.2C
            </P>
            <P>
                The material of this course is of the U.S Coast Guard's Rules of
the Road in
                compliance with U.S. Regulations.
            </P>
            </a>
            <BR>
        <a name="divisione">
            <P>
                <B>LESSON OBJECTIVE:</B> This course will give the student a
basic
                understanding of the Inland Rules of Navigation. These rules have
been Coast
                Guard approved according to the instruction listed above and U.S.
Law.
            </P>
            <BR>
            <B>Enjoy the course<B>
        </a>
```

<BODY>

Nell'esempio sono evidenziati i tag <INPUT> nascosti che contengono il testo da inviare all'oggetto prompt. Inoltre nel codice relativo alla visualizzazione della pagina sono evidenziati dei tag <a>, utili per avere dei riferimenti sul testo da evidenziare. Infatti si può notare che ci sono tre blocchi di testo e tre tag <a >.

Le varie funzioni implementate dal lettore sono attivate mediante le seguenti funzioni JavaScript :

```
        function startText()
        {
           if (elem == 0) {
             for (var i=0; i < document.anchors.length ; i++){
               document.anchors[i].style.backgroundColor = values[0];
               document.anchors[i].style.color = values[1];
```

```
      }
   }
   promptText.Start(document.formtext.elements[elem].value);
   document.anchors[elem].style.backgroundColor = values[1];
   document.anchors[elem].style.color = values[0];
   document.anchors[elem].scrollIntoView(true);
}

function nextText()
{
   if (elem < (document.formtext.length - 1)){
     elem = elem + 1;
     startText();
   }
   else
   {
     elem = 0;
   }
}

function stopText()
{
   promptText.Stop();
   elem = 0;
}

function continueText()
{
   promptText.Resume();
}

function pauseText ()
{
   promptText.Pause();
}
```

La funzione startText() è quella che effettua la lettura del blocco di testo corrente, passandolo come parametro alla funzione Start() dell'oggetto promptText. Il blocco di testo da eseguire viene reperito

dall'array degli elementi del form (formtext) contenente i vari <INPUT> ; per tener traccia del blocco corrente è utilizzato un contatore (elem). L'inizio della lettura è associato all'evento OnClick del buttonStart presente nella pagina Speech.htm; cliccando viene eseguita la funzione startText() sulla pagina del contenuto corrente.

La funzione startText() si occupa inoltre di evidenziare il blocco di testo corrente invertendo i colori di carattere e background, e di far scorrere la pagina fino al blocco corrente.

La funzione nextText(), viene invocata alla fine dell'esecuzione del prompt, per poter passare al blocco di testo successivo; viene incrementato il contatore e richiamata la funzione startText(). Nella definizione dell'elemento prompt della pagina (<salt:propmpt>), si può notare che all'evento OnComplete è associata l'esecuzione della funzione nextText().

Le funzioni stopText(), continueText(), pauseText(), eseguono rispettivamente le funzioni Stop(), Resume() e Pause(), sull'oggetto promptText.

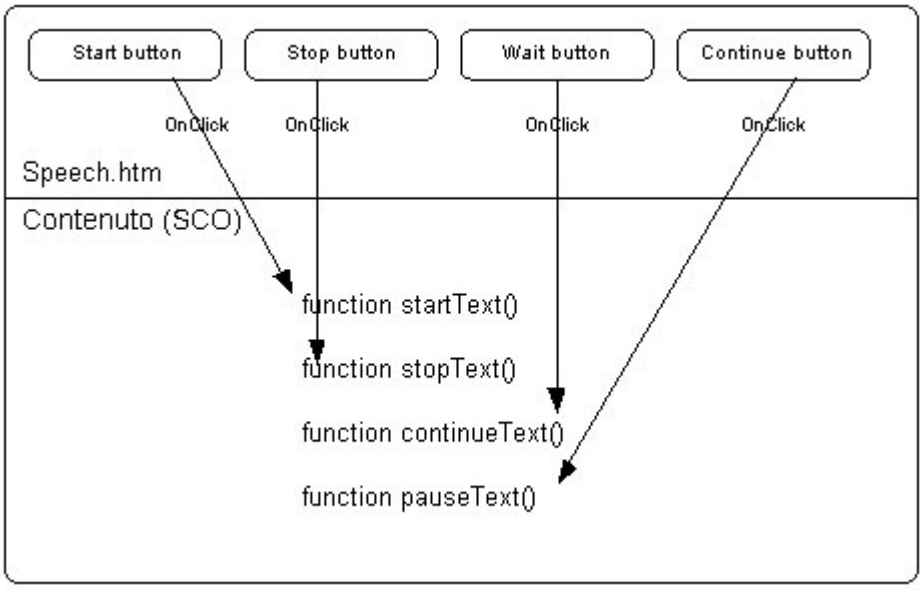

Figura 28: Schema delle chiamte a funzione del lettore

La struttura è stata pensata in modo da permettere in futuro, lo sviluppo di un sistema di trascodifica automatica dei contenuti. Le funzioni JavaScript per la gestione dell'output vocale e l'elemeto prompt per l'esecuzione dell'output vocale sono indipendenti dai contenuti; sono invece dipendenti dal contenuto il testo nascosto relativo al testo da leggere e i punti di interruzione per la notifica dello stato

di avanzamento. Lo sviluppo di un motore di trascodifica non dovrebbe quindi presentare grosse difficoltà.

5.3.3 Comandi vocali del visualizzatore

Come già accennato nel paragrafo 5.2.1, ci sono alcuni comandi vocali abilitati solo per la fase di fruizione del corso :

- Back : contenuto precedente;
- Next : contenuto successivo;
- Close : chiusura corso;
- Start : inizio lettura contenuto;
- Stop : fine lettura contenuto;
- Wait : sospensione lettura contenuto;
- Continue : proseguimento lettura contenuto.

Questi comandi possono essere riconosciuti dal componente listen presente nella pagina Upperbar.aspx che implementa il menu dell'applicazione. Grazie a funzioni JavaScript è possibile far riferimento ai vari frame che contengono gli elementi interessati dall'esecuzione del comando.
L'esempio seguente mostra la funzione associata al comando Start :

```
function commandStart(){
  if (parent.frames("Frame_content").document.title == "Content"){
    wincenter = parent.frames("Frame_content").frames("center");
    if (wincenter.document.formspeech != null){
        wincenter.startText();
    }
  }
}
```

Come si può vedere grazie all'oggetto parent è possibile far riferimento al frame center (che contiene la pagina del contenuto) partendo dalla pagina UpperBar.aspx. Una volta individuata la pagina del contenuto (wincenter), viene eseguito il metodo startText().

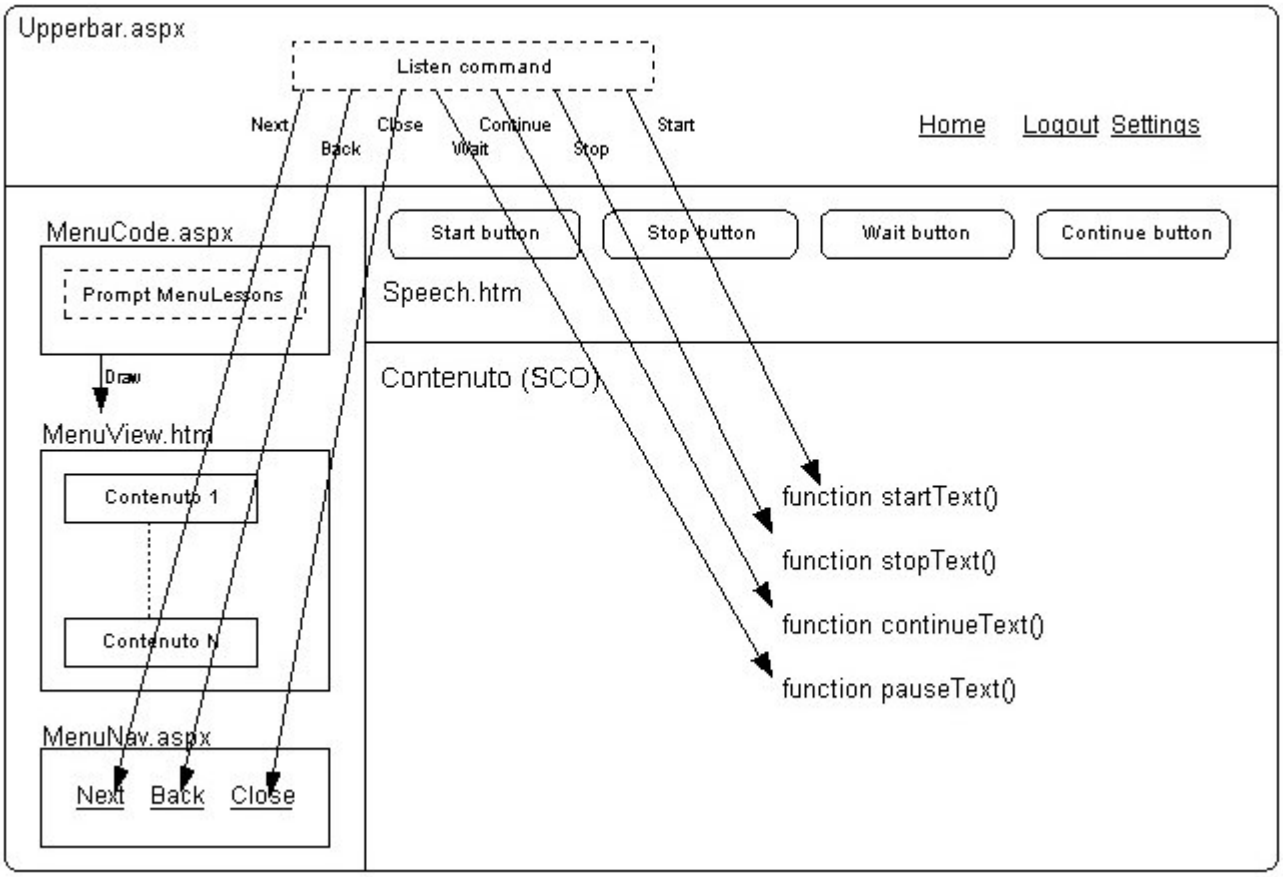

Figura 29: Schema dei comandi vocali del visualizzatore

5.4 Configurazione contrasto/carattere

Le funzionalità di configurazione dell'applicazione sono implementate dalla pagina SetStyle.aspx caricabile dal link Settings della pagina UpperBar.aspx.

I valori che possono essere impostati in questa pagina sono il colore del background (combobox con colori predefiniti), il colore del testo (combobox con colori predefiniti) e la dimensione del carattere (textbox numerico con bottoni per incrementare/decrementare). A ciascun campo è associato un componente prompt per la guida vocale della pagina.

Lo stile degli elementi delle pagine dell'applicazione è definito nel file mainstyle.css (Component Style Sheet); ogni pagina dell'applicazione contiene un riferimento a questo foglio di stile. I valori di default sono background bianco, carattere nero e dimensione 25pt .

Le varie modifiche apportate dall'utente vengono memorizzate in un cookie nella forma :

```
<background color>|<font color>|<font size>
```

ed impostate nella fase di caricamento delle varie pagine. Per fare tutto ciò sono state sviluppate delle funzioni JavaScript in grado di intervenire sugli stili della pagina.
Le funzioni per l'impostazione degli eventuali valori modificati sono le seguenti :

```
function setActiveStyleSheet(mlstyle) {
  var values = mlstyle.split('|');
  if (values[0] != ""){
     setBackground(values[0]);
  }
  if (values[1] != ""){
     setColor(values[1]);
  }
  if (values[2] != ""){
     setFont(values[2]);
  }
}

function setBackground(value){
  stile = document.styleSheets[0].rules;
  stile[0].style.backgroundColor = value;
}

function setColor(value){
  stile = document.styleSheets[0].rules;
  for (var i=0; i < stile.length ; i++){
    stile[i].style.color = value;
  }
}

function setFont(value){
  stile = document.styleSheets[0].rules;
  for (var i=0; i < stile.length ; i++){
    stile[i].style.fontSize = value;
  }
}
```

Alla funzione setActiveStyleSheet() viene passato come parametro un oggetto contenete il cookie, dal quale vengono estratti i valori da passare alle funzioni specifiche setBackground(), setColor(),

setFont(). Grazie all'array styleSheets dell'oggetto document, è possibile ottenere tutte le regole di stile definite nel file mainstyle.css, e modificarne i valori con quelli salvati nel cookie.

Una caratteristica fondamentale è che la modifica di un valore deve essere immediatamente applicata allo stile della pagina SetStyle.aspx, per dare all'utente la possibilità di valutare eventuali miglioramenti di percezione visiva. In questo caso alla modifica di un valore, viene chiamata direttamente una delle tre funzioni setBackground(), setColor() o setFont().

6 ESEMPI D'USO E VALUTAZIONE DEI RISULTATI

6.1 Autenticazione

La procedura si autenticazione avviene tramite l'immissione da parte dell'utente delle p roprie credenziali, utilizzando l'interfaccia mostrata nella figura segeuente :

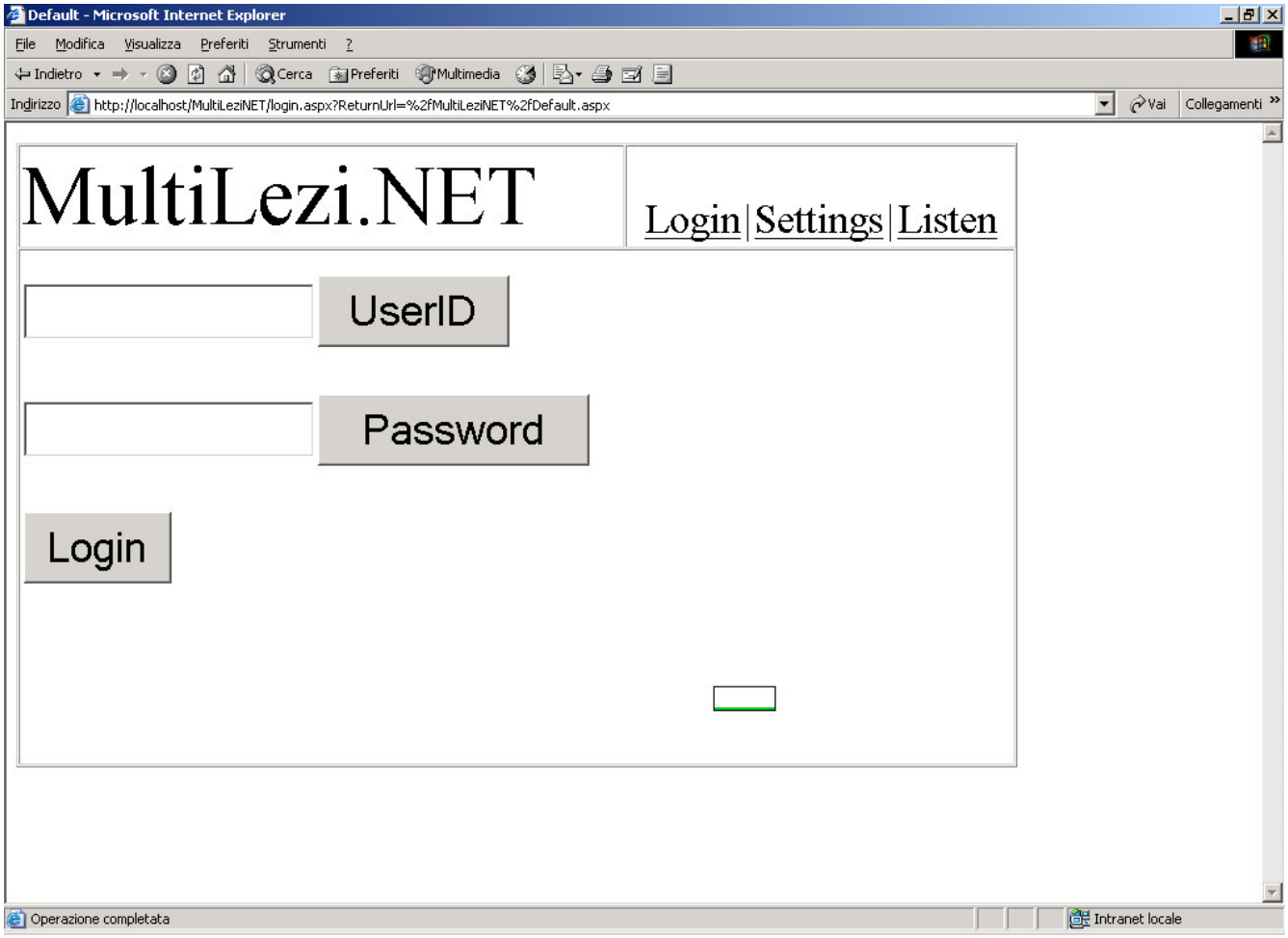

Figura 30: Pagina di autenticazione, Login.aspx

Come si può vedere, l'interfaccia è semplice ed essenziale, in accordo con le specifiche stabilite. I button UserID e Password servono per il riconoscimento vocale rispettivamente di userID e password, mentre il button Login esegue l'accesso all'applicazione. I link in alto servono per passare dalla pagina di login (Login) a quella di configurazione (Settings) e viceversa, e per riattivare il riconoscimento dei

comandi vocali (Listen). Il riconoscimento vocale, come si può vedere, è segnalato da un piccolo rettangolo che traccia l'andamento degli input vocali. In questo caso si tratta del riconoscimento dei comandi vocali.

La figura seguente invece mostra la fase di inserimento vocale della password, in seguito all'inserimento dello userID. Si può notare la differenza del grafico dell'input vocale, rispetto alla Figura 30 che ritrae una situazione di attesa.

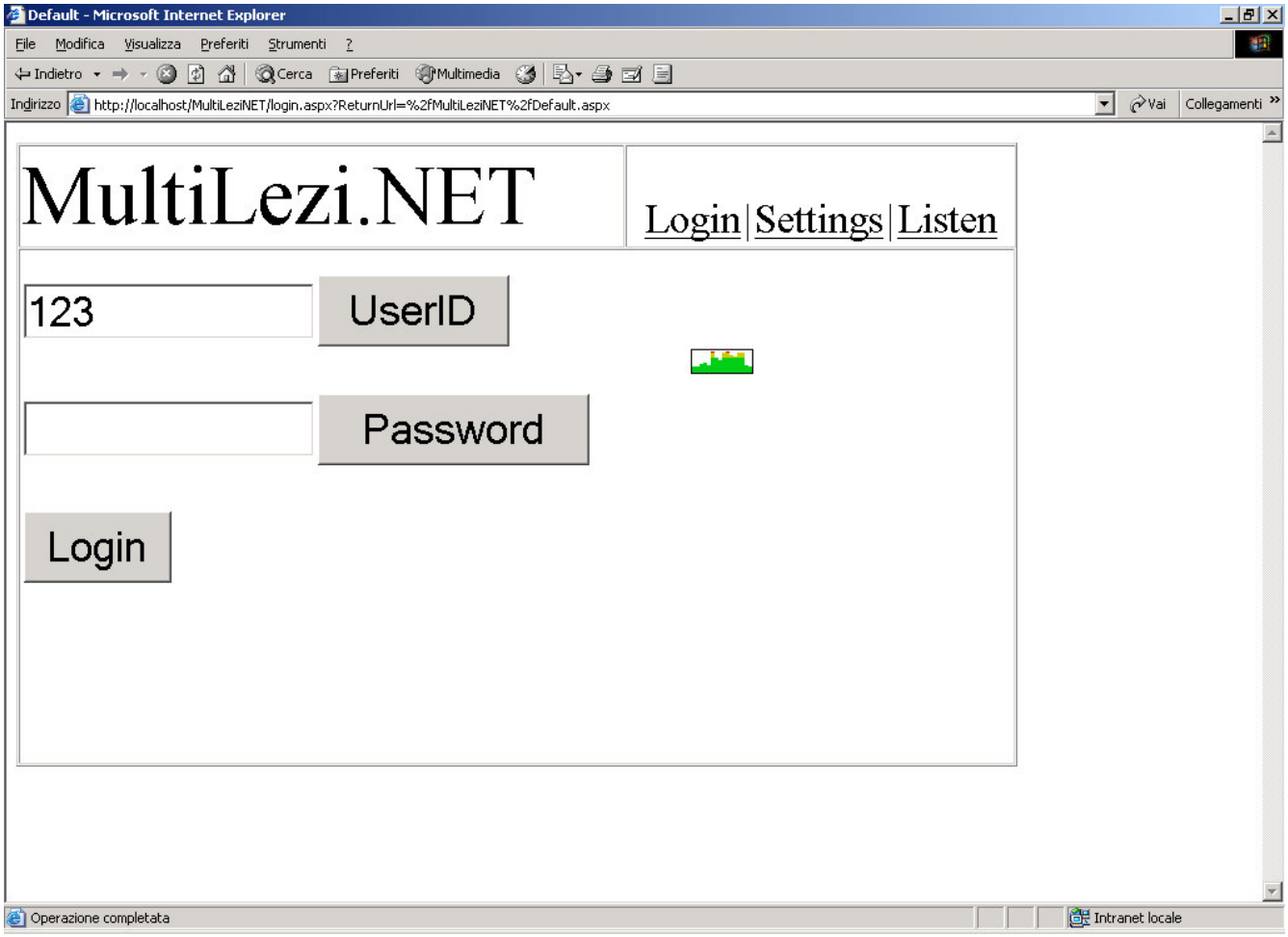

Figura 31: Fase di inserimento vocale

Il contrasto e la dimensione del carattere sono quelli di default; in seguito saranno mostrate altre impostazioni.

6.2 Consultazione corsi

Una volta ottenuto l'accesso all'applicazione, la pagina che si presenta è la seguente :

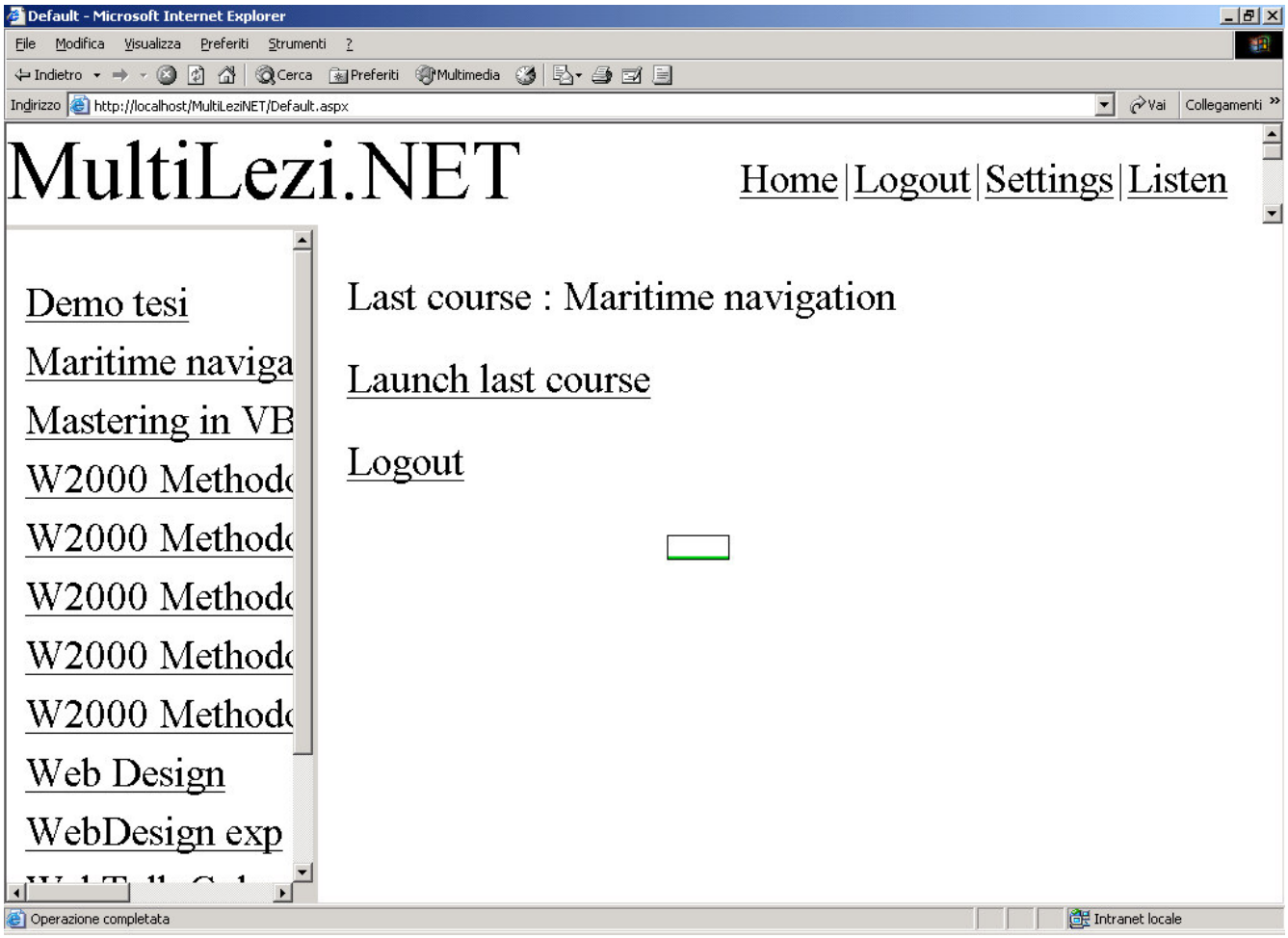

Figura 32: Pagina principale dell' applicazione

E' la pagina principale dell'applicazione che permette la consultazione dei corsi e dei relativi argomenti.

Il frame in alto contiene il menù generale, con link che permettono di tornare alla Homepage (Home), uscire dall'applicazione (Logout), passare alla pagina di configurazione (Settings) e riattivare il riconoscimento vocale (Listen).

Il frame di sinistra presenta la lista dei corsi disponibili, mentre il frame centrale segnala l'ultimo corso lanciato, il link del corso, ed un link per il logout. Selezionando un corso, il frame centrale presenterà la lista dei relativi argomenti.

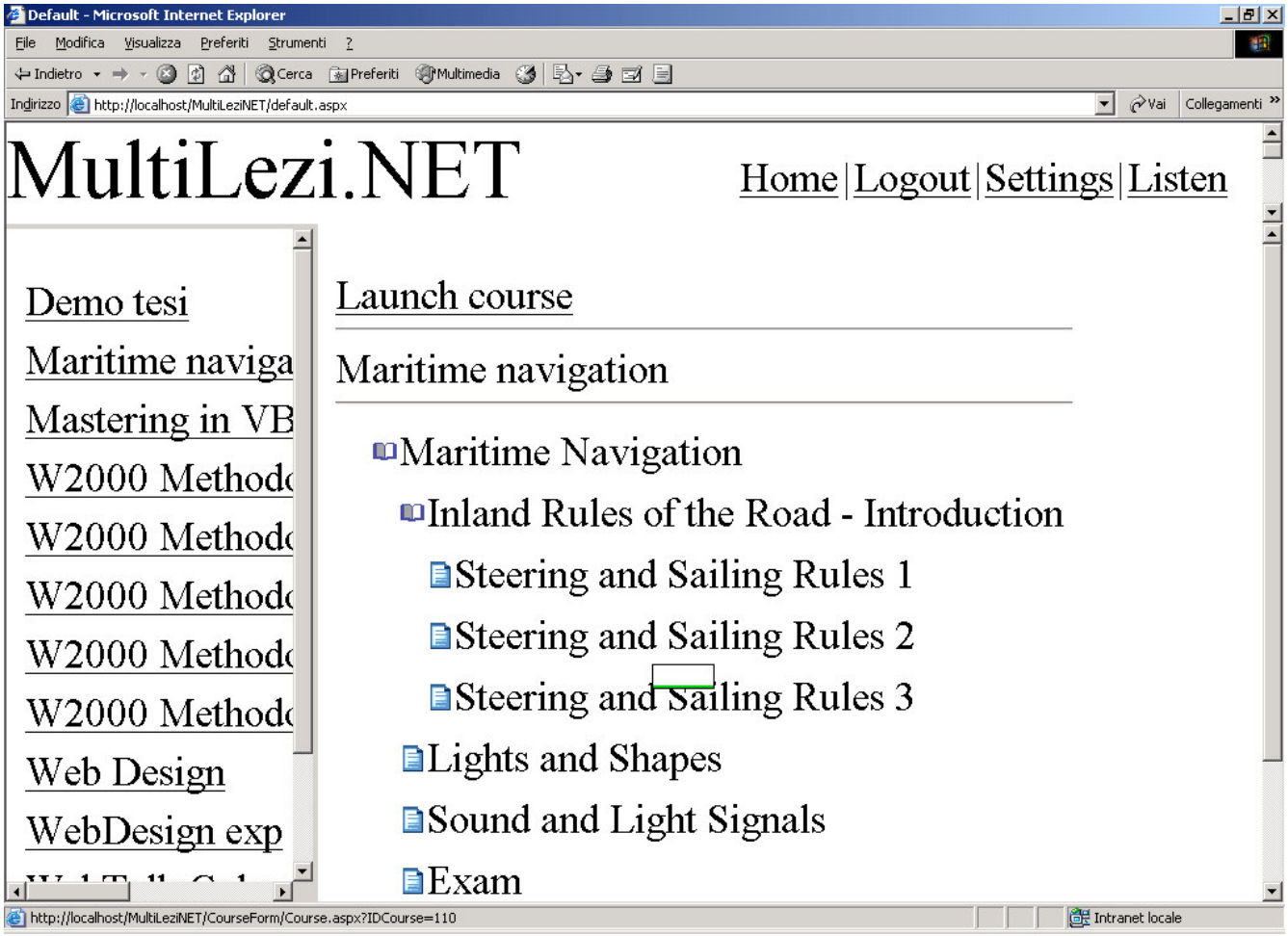

Figura 33: Selezione di un corso

La lista degli argomenti è preceduta dal link che permette di lanciare il corso, caricando la pagina del visualizzatore.

La guida vocale permette di scoprire i contenuti navigando con il mouse nelle varie zone della pagina, ed esistono dei comandi vocali che facilitano l'esecuzione delle funzioni base, come l'uscita dall'applicazione o il lancio dell'ultimo corso. Per una descrizione dettagliata dell'interfaccia vocale fare riferimento al capitolo 5.

6.3 Fruizione corso

La pagina che si presenta per la fruizione di un corso è la seguente :

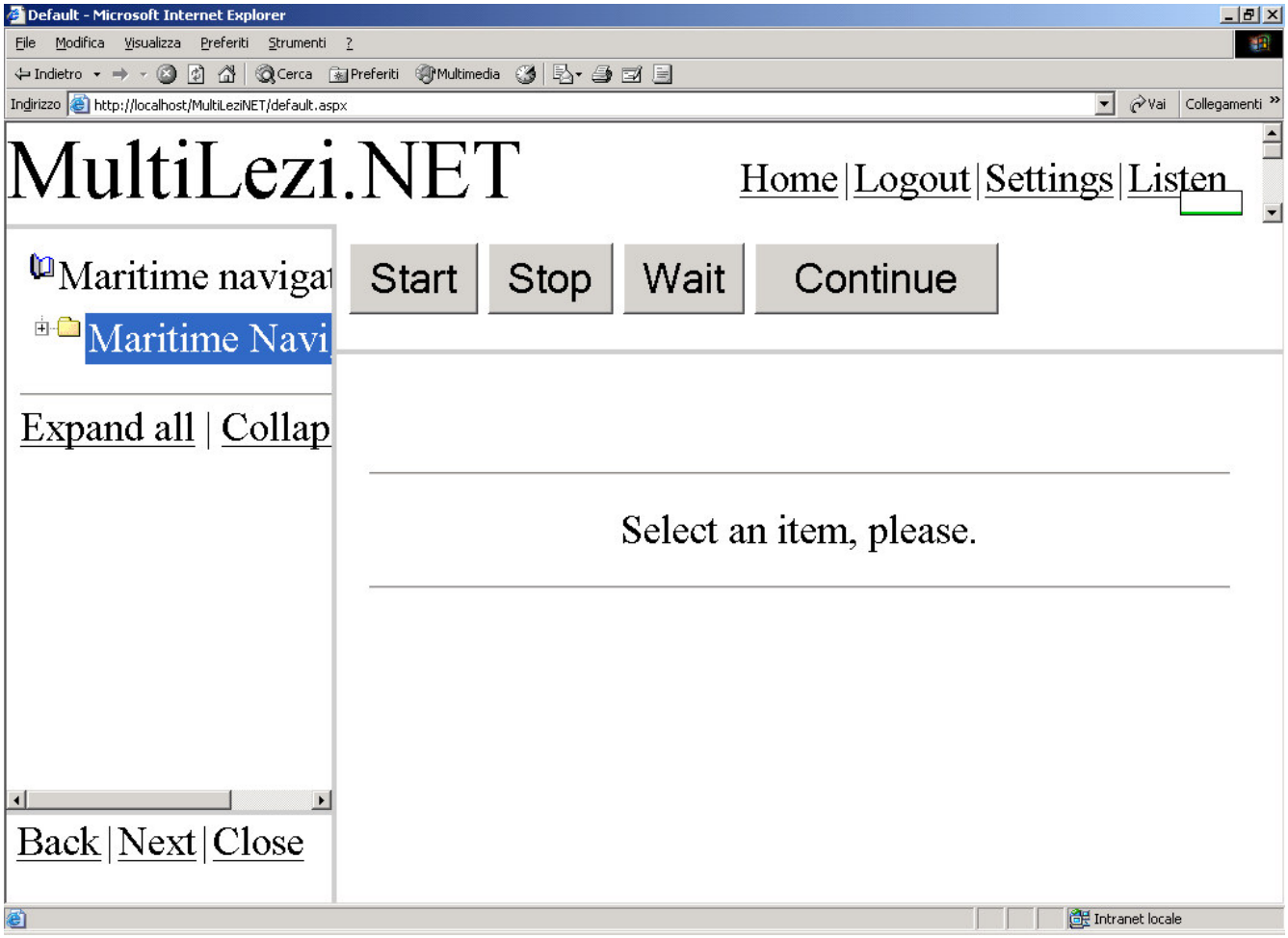

Figura 34: Pagina per la fruizione dei corsi

Il frame di sinistra presenta un menù ad albero con i contenuti del corso selezionato, mentre il frame centrale è composto dall'area che sarà occupata dai contenuti selezionati e dai pulsanti per la gestione della lettura del contenuto. In basso a sinistra ci sono dei link per facilitare la navigazione tra i contenuti del corso e per la chiusura del corso. Esplorando la pagina con il mouse, la guida vocale permette di riconoscere i vari componenti della pagina. Sono inoltre disponibili dei comandi vocali per la gestione della lettura, per la navigazione da un contenuto all'altro e per la chiusura del corso. Infatti in alto a sinistra si può notare il grafico del riconoscimento vocale.

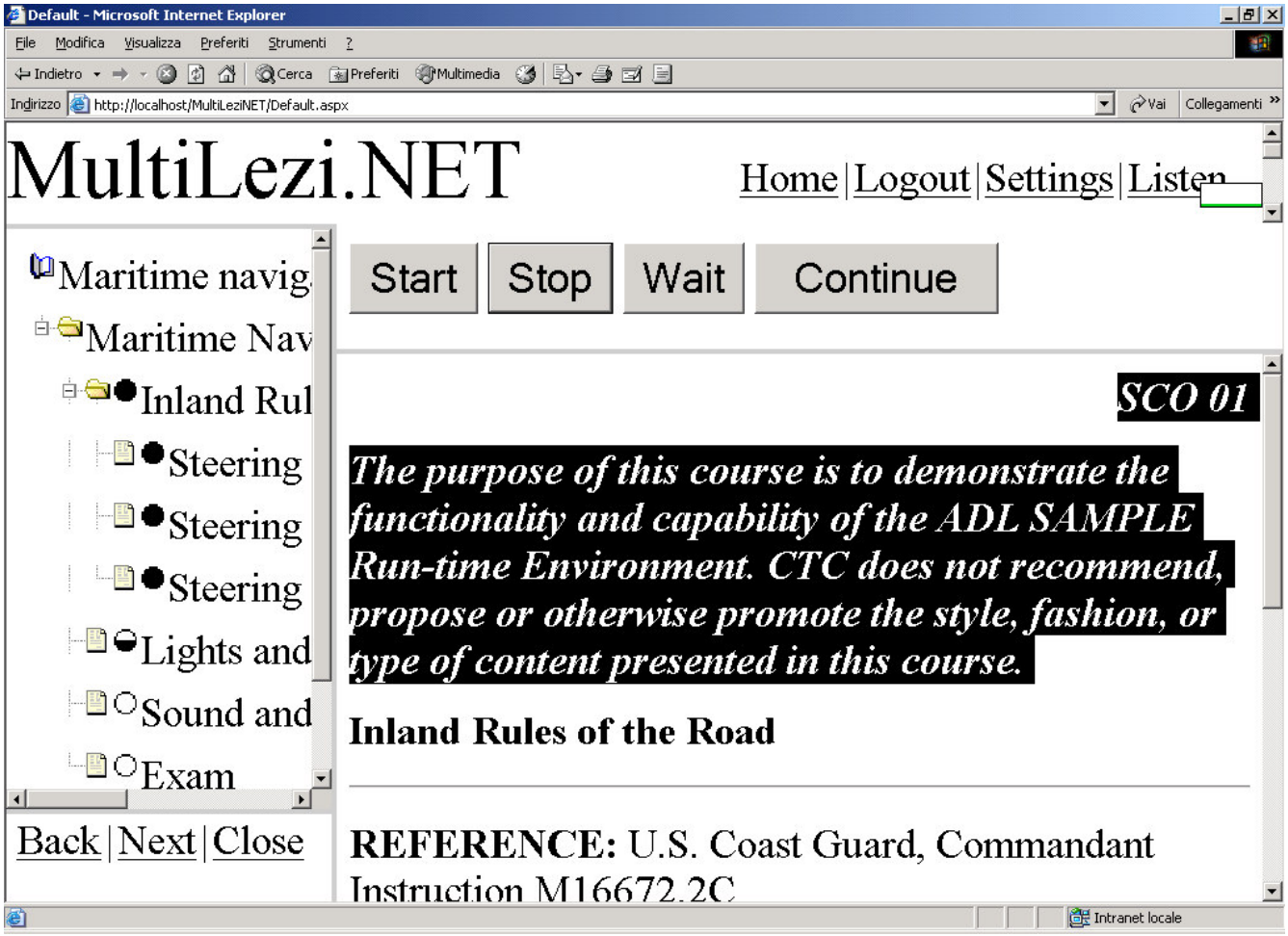

Figura 35 : Lettura di un corso con stato di avanzamento

Navigando nell'albero dei contenuti è possibile vedere lo stato di ogni contenuto : un pallino bianco indica che il contenuto non è ancora stato visualizzato, un pallino metà nero e metà bianco indica che il contenuto è stato visualizzato ma l'attività di apprendimento non è ancora stata conclusa, mentre un pallino nero indica che il contenuto è stato appreso.

Durante la lettura del contenuto (che può essere attivata con il pulsante start o il comando start), lo stato di avanzamento è evidenziato rispetto al resto del testo.

6.4 Configurazione applicazione

La pagina di configurazione è attivabile dal link Settings presente nella parte superiore di ogni pagina dell'applicazione. Essa permette di impostare colore dello sfondo, colore del carattere e dimensione del carattere.

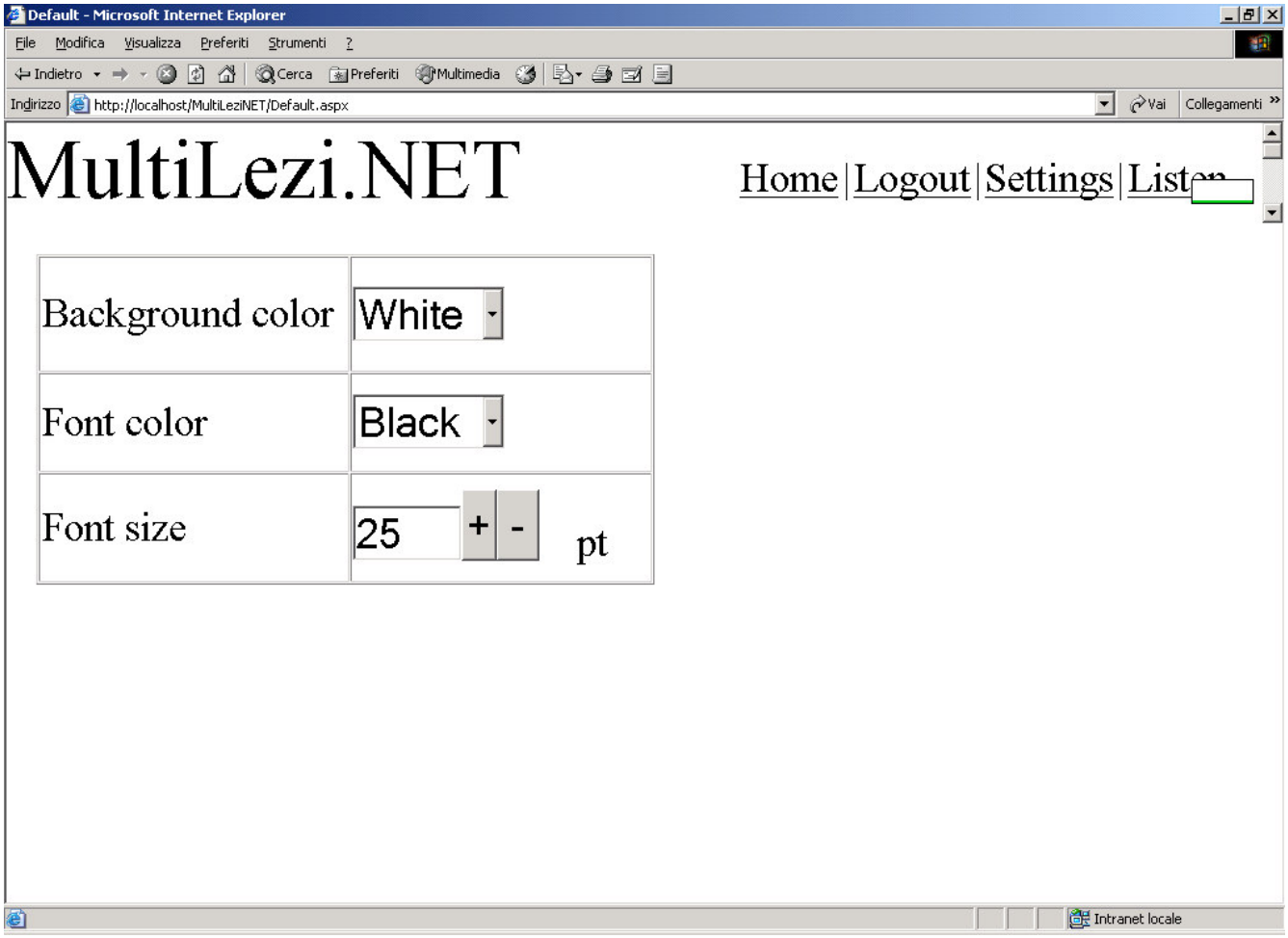

Figura 36 : Pagina di configurazione

Cambiando uno dei parametri di configurazione, la modifica viene immediatamente applicata alla pagina, semplificando così le scelte da effettuare. Lasciando la pagina di configurazione, le ultime impostazioni sono applicate a tutte le altre pagine dell'applicazione.
Alcuni esempi di configurazione sono mostrati di seguito :

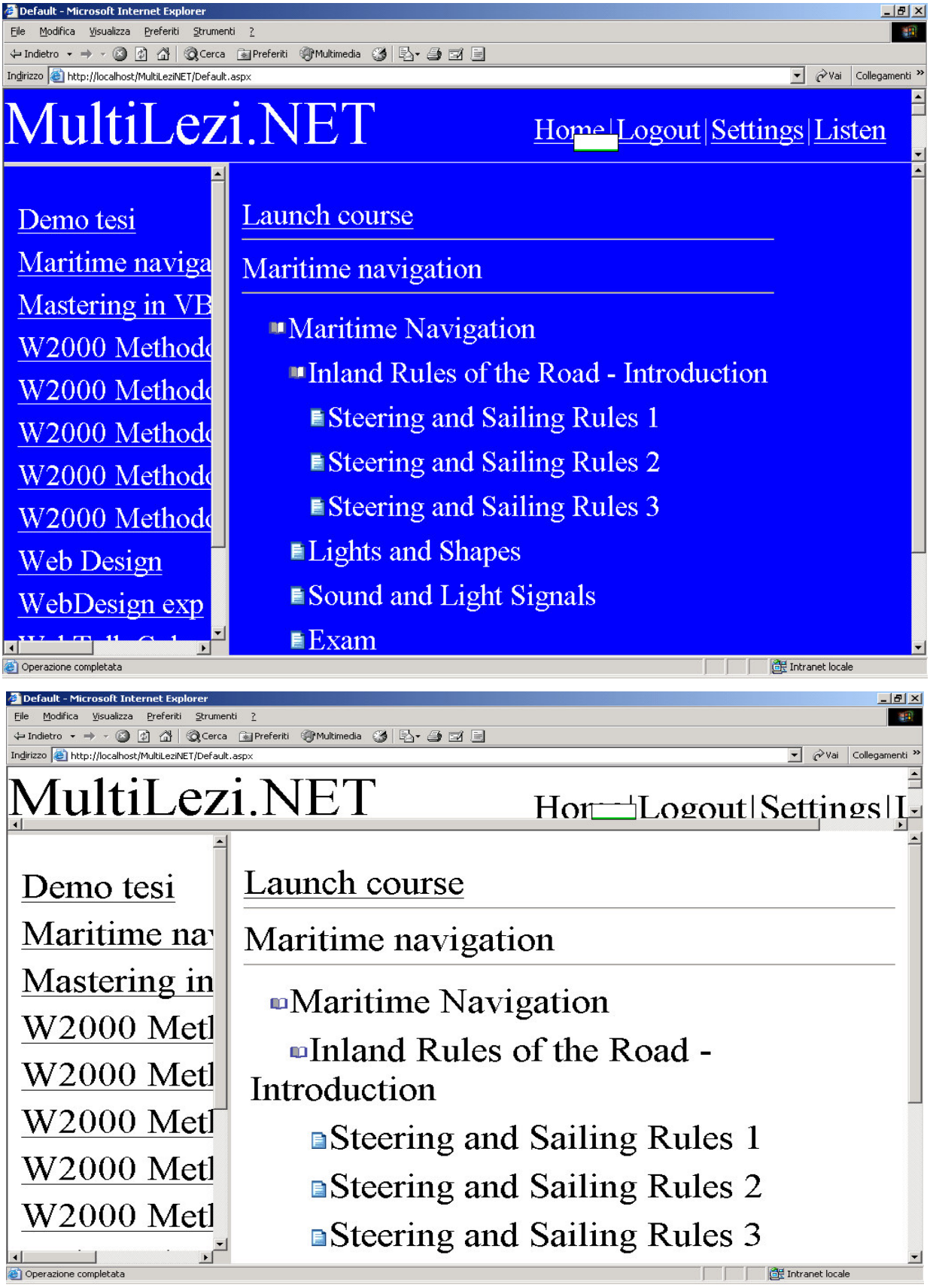

Figura 37: Esempi di configurazione colore e carattere della Homepage

6.5 Valutazione dei risultati ottenuti

Al fine di valutare la reale efficacia di MultiLezi.NET, sono stati condotti dei test coinvolgendo utenti ipovedenti. In particolare ad un gruppo di dieci utenti è stato chiesto di assimilare il primo contenuto di un determinato corso. Prima del test gli utenti sono stati informati sulle funzionalità dell'applicazione e sulle modalità d'interazione dell'applicazione attraverso una breve demo; in particolare sono stati spiegati loro i comandi vocali disponibili. Quindi partendo dalla pagina di login sono state fornite loro delle credenziali di test ed è stato chiesto loro di raggiungere il contenuto "Inland rules of the road" del corso "Maritime navigation", e di assimilarne i contenuti.

L'obiettivo principale era quello di verificare che gli utenti riuscissero a raggiungere effettivamente il contenuto richiesto. Altre metriche di valutazione prese in considerazione sono state :

- Utilizzo delle modalità : desiderio di valutare l'utilizzo delle due modalità di interazione (vocale e manuale);
- Tempi di esecuzione delle attività richieste : desiderio di avere un riscontro sui tempi impiegati per portare a termine l'attività richiesta
- Modifica delle impostazioni : desiderio di avere un riscontro sull'utilizzo e l'efficacia di tecniche adattative, quali la modifica del contrasto e l'ingrandimento del carattere

I risultati ottenuti nei test sono riassunti nella tabella seguente :

Utenti	**Fruizione del contenuto**	**Modalità utilizzate**	**Tempi di esecuzione**	**Modifica impostazioni**
Utente 1	SI	Manuale/Vocale	5 minuti	Contrasto/Carattere
Utente 2	SI	Manuale	15 minuti	Contrasto/Carattere
Utente 3	SI	Manuale/Vocale	8 minuti	Carattere
Utente 4	SI	Manuale/Vocale	10 minuti	Carattere
Utente 5	SI	Vocale/Manuale	8 minuti	-
Utente 6	SI	Manuale/Vocale	10 minuti	Contrasto/Carattere
Utente 7	SI	Manuale	7 minuti	Carattere
Utente 8	SI	Vocale/Manuale	12 minuti	-
Utente 9	SI	Manuale/Vocale	9 minuti	Contrasto/Carattere
Utente 10	SI	Vocale/Manuale	6 minuti	-

Come si può notare, tutti gli utenti sono riusciti a portare a termine l'attività che era stata loro chiesta, dimostrando che l'applicazione risulta essere utilizzabile.

Per quanto riguarda le modalità d'interazione sono state classificate nel modo seguente :

- Manuale : l'utente non ha utilizzato né comandi vocali, né il lettore dell'applicazione;
- Manuale/Vocale : l'utente ha utilizzato prevalentemente la modalità manuale (tastiera e mouse) e saltuariamente comandi vocali ed il lettore;
- Vocale/Manuale : l'utente ha utilizzato prevalentemente comandi vocali ed il lettore dell'applicazione, e saltuariamente la modalità manuale.

L'interazione Manuale/Vocale è stata quella più utilizzata ed in particolare l'utilizzo di comandi vocali è stato scelto in presenza di difficoltà nell'esecuzione di determinate funzioni; il lettore è stato utilizzato invece da utenti con grosse difficoltà a leggere anche con grandi dimensioni di carattere e contrasto adeguato.

L'interazione unicamente Manuale è stata invece utilizzata da utenti in grado di leggere con adeguate impostazioni di carattere e contrasto.

L'utilizzo dell'interazione Vocale/Manuale, è stata invece intrapresa da utenti desiderosi di sperimentare nuove modalità di interazione; l'utilizzo della modalità manuale è stato scelto in alcuni punti dell'applicazione che non prevedono comandi vocali, come ad esempio la scelta del corso.

I tempi di esecuzione sono abbastanza elevati, anche se bisogna tener conto del fatto che gli utenti non conoscevano l'applicazione; un tempo medio di esecuzione pari a 9 minuti è comunque accettabile considerando le difficoltà degli utenti.

Per quanto riguarda la modifica delle impostazioni di contrasto e carattere, si può notare che sono state utilizzate principalmente nelle interazioni prevalentemente manuali; in questi casi occorre comunque segnalare che la modifica delle configurazioni è stata quasi sempre determinante per la fruizione del contenuto.

Sebbene i test abbiano dato dei risultati positivi, sono anche serviti ad evidenziare gli aspetti da migliorare come:

- Miglioramento dell'interfaccia della pagina di configurazione : comandi vocali per aumentare/diminuire il carattere o per selezionare il colore;
- Aggiunta di un help vocale e visuale per la descrizione dei comandi vocali disponibili;
- Lettura sequenziale della lista dei corsi e degli argomenti del corso.

CONCLUSIONI E SVILUPPI FUTURI

A conclusione della realizzazione di MultiLezi.NET, si può affermare che tutti i requisiti sono stati soddisfatti.
L'obiettivo principale di realizzare un'applicazione di e-learning per utenti ipovedenti è stato raggiunto, e a giudicare dai risultati ottenuti nei test, con esito positivo. Inoltre l'integrazione di MultiLezi.NET nella piattaforma Lezi.NET è avvenuta con successo, in quanto la sincronizzazione degli accessi è stata mantenuta. Per quanto riguarda la tecnologia SALT, si può dire che ben si presta allo sviluppo di interfacce multimodali e che è stata di fondamentale importanza per l'implementazione di MultiLezi.NET.

Per quanto riguarda eventuali sviluppi futuri, si può pensare di migliorare l'interfaccia multimodale aggiungendo altre modalità d'interazione e di estendere le funzionalità di MultiLezi.NET.
L'utilizzo di mouse gestures per l'esecuzione di alcuni comandi o funzioni potrebbe essere una buona idea, come anche l'adozione di mouse tattili in grado di emettere vibrazioni per comunicare informazioni.
Un aspetto di sicuro interesse riguarda lo sviluppo di un sistema per la trascodifica automatica dei contenuti. Non a caso la struttura dei contenuti multimodali del corso d'esempio "Maritime navigation", è stata pensata nell'ottica di permettere in futuro lo sviluppo di un sistema di questo tipo.

BIBLIOGRAFIA

[1] ADL - SCORM : Advanced Distribuited Learning – Sharable Content Object Reference Model : *SCORM Overview, SCORM Content Aggregation Model, SCORM Run-Time Environment*. URL : http://www.adlnet.org .

[2] T. Barbieri, A. Bianchi e L. Sbattella. *Multimodal communication for vision and hearing impairments*. set. 2004.

[3] C# International Standard. URL : http://www.ecma-international.org .

[4] E-LEARNING : URL : http://www.epsilonlearning.com ; http://www.wbt.it ; http://www.tecnoteca.it/tesi/e_learning/ ; http://www.elearningeuropa.info ; http://www.elearningtouch.it .

[5] ETSI : European Telecomunications Standard Institute. *Human Factors (HF); Multimodal interaction, communication and navigation guidelines*.

[6] Judy Brewer. *How People with Disabilities Use the Web*. URL : http://www.w3.org/WAI/EO/Drafts/PWD-Use-Web/ .

[7] IEEE LTSA : Learning Technology Systems Architecture : *Draft standard for learning technology*. URL : http://edutool/ltsa

[8] IMS Content Packaging specification. URL : http://www.imsglobal.org .

[9] Scott McGlashan, Daniel C. Burnett, Jerry Carter, Peter Danielsen, Jim Ferrans, Andrew Hunt, Bruce Lucas, Brad Porter, Ken Rehor, Steph Tryphonas. *Voice Extensible Markup Language (VoiceXML)*. URL: http://www.w3.org/tr/voicexml20/ .

[10] Stéphane H. Maes, Vijay Saraswat. *Multimodal Interaction Requirements*. URL : http://www.w3.org/TR/mmi-reqs/ .

[11] MAIS : Multichannel Adaptive Information System. URL : http://black.elet.polimi.it .

[12] Microsoft ASP.NET. URL : http://www.asp.net .

[13] Microsoft .NET Framework. URL : http://msdn.microsoft.com/framework .

[14] T.V. Raman. *User interface principle for multimodal interaction*. CHI 2003.

[15] T. V. Raman, Gerald McCobb, Rafah A. Hosn. *XHTML+Voice 1.1 Versatile Multimodal Solutions*. XML Journal, apr. 2003.

[16] Leah M. Reeves, Jennifer Lai, James A. Larson, Sharon Oviatt, T.S. Balaji, Stéphanie Buisine, Penny Collings, Phil Cohen, Ben Kraal, Jean-Claude Martin, Michael McTear, TV Raman, Kay

M. Stanney, Hui Su, QianYing Wang. *Guidelines for multimodal user interface design*. Communications of the ACM – Special Issue on Multimodal Interfaces, vol. 47(1), pp. 57-59, 2004.

[17] SALT : Speech Application Language Tags: *SALT technical white paper, SALT specification 1.0* . URL : http://www.saltforum.org .

[18] SASDK : Microsoft Speech Application SDK. URL : http://www.microsoft.com/speech .

[19] VoiceXML : Voice eXstensible Markup Language. URL : http://www.voicexml.org .

[20] W3C Multimodal Interaction Activity. URL : http://www.w3.org/2002/mmi/ .

[21] W3C SOAP : Simple Object Access Protocol. URL : http://www.w3.org/TR/2003/PRsoap12-part1-20030507/ ; http://www.w3.org/TR/2003/PR-soap12-part2-20030507/ .

[22] W3C UDDI : Universal Description, Discovery and Integration. URL : http://www.oasisopen.org/committees/uddi-spec/doc/tcspecs.htm .

[23] W3C WSDL : Web Services Description Language. URL : http://www.w3.org/TR/wsdl12/ ; http://www.w3.org/TR/wsdl12-bindings/

[24] Kuansan Wang. *SALT: a spoken language interface for web-based multimodal dialog systems*. Speech Technology Group, Microsoft Research.