Consiglio Nazionale delle Ricerche

Nota Interna

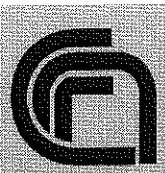
Il disegno e la realizzazione di un prototipo di interfaccia web accessibile

Vittorio Miori, Gabriele Tolomei

B4-22
dic-2002

ISTI
ISTITUTO DI
SCIENZA E TECNOLOGIE DELLA
INFORMAZIONE "Alessandro Faedo"

Pisa



Consiglio Nazionale delle Ricerche

Il disegno e la realizzazione di un prototipo di interfaccia web accessibile

Vittorio Miori, Gabriele Tolomei

B4-22
dic-2002

Il disegno e la realizzazione di un prototipo di interfaccia web accessibile

Vittorio Miori
Gabriele Tolomei

ISTI B4-23
Dicembre 2002

INDICE

INTRODUZIONE

Il documento tratta della realizzazione di un'interfaccia accessibile agli utenti disabili per il sistema informativo INFEA: INFEA [a] costituisce un significativo esempio di *WIS (Web-based Information System)* dedicato alla Pubblica Amministrazione, che coinvolge le iniziative rivolte allo Sviluppo Sostenibile del Ministero dell'Ambiente.

Si rimanda al documento ISTI B4-22, per le ulteriori informazioni e approfondimenti riguardo sia le problematiche specifiche sull'accessibilità e usabilità dei siti Web, sia per gli approfondimenti rivolti allo studio delle tecnologie utilizzate.

Il sistema INFEA nasce per permettere agli utenti di consultare e ricercare informazioni messe a disposizione dal Ministero nell'ambito dell'iniziativa rivolta allo Sviluppo Sostenibile; in particolare è possibile avere come obiettivo di ricerca gli Enti coinvolti, i Servizi offerti, le Manifestazioni e molte altre tipologie di ricerca.

Il prototipo di interfaccia sviluppato secondo le norme di accessibilità e usabilità web riguarda la categoria Enti, che rappresenta la porzione di informazione più importante contenuta nel sistema; le modifiche che sono state necessarie per adeguare l'attuale interfaccia a quella accessibile sono comunque facilmente replicabili per le restanti categorie di ricerca.

Come consigliato dalle norme seguite, il codice *JavaScript*, ampiamente impiegato nella costruzione dell'interfaccia attualmente in servizio, è stato del tutto eliminato, intervenendo su 3 componenti del sistema: codice HTML, Servlet Java e fogli di stile XSL.

Per quel che riguarda il codice HTML, è stata notevolmente snellita la home page, eliminando molte immagini di scarsa utilità ai fini della navigazione. L'intervento più consistente, tuttavia, è stato attuato sui documenti dedicati all'impostazione dei filtri di ricerca. Inoltre, le informazioni sono state riorganizzate secondo un criterio di accesso sequenziale, necessario agli utenti disabili, e in particolar modo a quelli con disturbi a livello visivo.

---

[a] http://www.infea2001.cnuce.cnr.it/infea2001/home.html

È stato necessario creare due nuove Servlet Java per gestire la ricezione di tutti i parametri di un'interrogazione provenienti da un unico form, e quindi costruire il file XML rappresentante l'interrogazione e modificare il file XML contenente i risultati di una ricerca.

Per poter seguire la stessa impostazione scelta per la visualizzazione di documenti HTML, anche i fogli di stile XSL hanno subito alcune modifiche; le pagine dinamiche XHTML così ottenute hanno, come fonte dei dati, il file XML contenente i risultati delle interrogazioni.

Una volta terminate le modifiche, è stato necessario effettuare una serie di test di validazione del prototipo, attraverso l'ausilio di tools automatici via web (come *Bobby* [c] di CAST) e grazie alla preziosa collaborazione di una persona disabile, che attualmente sta svolgendo dottorato di ricerca presso l'Istituto ISTI del CNR di Pisa.

I test sono stati effettuati per verificare la funzionalità del prototipo sia nell'ambiente su cui è stato sviluppato (PC con sistema operativo Windows 2000 NT Professional), sia su altre postazioni (Macintosh con sistema operativo Mac OS 9.1).

In entrambi i casi si sono utilizzati i due browser grafici più diffusi, Microsoft Internet Explorer (versione 5.0) e Netscape Navigator (versione 4.76); inoltre, per conoscere il comportamento del prototipo con un browser non grafico, si è aggiunta un'ulteriore fase di test attraverso l'impiego di Lynx [d] (versione 2.7.1 per Macintosh e 2.8.3 per Windows), user agent testuale sufficientemente usato sia da utenti disabili, sia nei paesi in cui lo sviluppo tecnologico è più arretrato.

Sulla piattaforma PC con sistema Windows 2000 NT Professional è stato effettuato un ulteriore test utilizzando Mozilla.

Il risultato ottenuto, pur rappresentando solo una parte del lavoro complessivo necessario a rendere l'intero sistema INFEA accessibile, costituisce un valido modello su cui basarsi per replicare le modifiche a tutte le informazioni che INFEA mette a disposizione dell'utente.

---

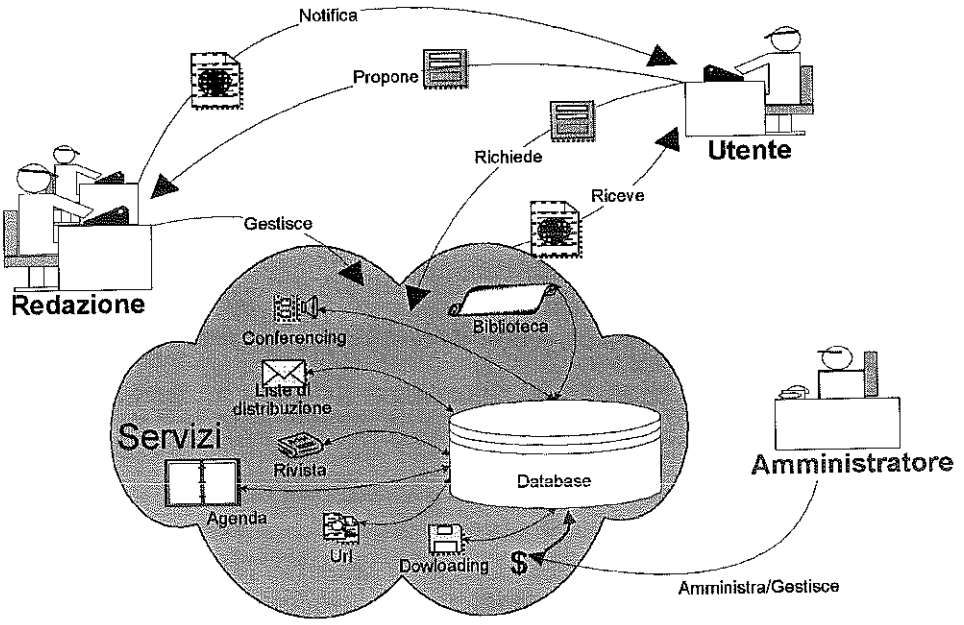
[c] http://bobby.watchfire.com/bobby/html/en/index.jsp

[d] http://lynx.browser.org

1. IL SISTEMA INFEA [4]

Il sistema informativo INFEA è un esempio di WIS realizzato per la Pubblica Amministrazione, nel caso specifico per il Ministero dell'Ambiente.

L'obiettivo di INFEA è di fornire, in maniera integrata, sia strumenti di consultazione, sia strumenti di gestione delle varie sorgenti di informazioni, con la possibilità di fruire le offerte informative degli Enti coinvolti nel sistema (come ad esempio l'ISFOL), i centri regionali di Educazione Ambientale, i laboratori territoriali e lo stesso Ministero.



- FIGURA 1.2 -

La figura 1.2 mostra uno schema ad alto livello del sistema INFEA, composto da un insieme di servizi che accedono al database tramite richieste inviate da diverse tipologie di utenti, ognuna avente ruoli e permessi differenti.

Tutti gli oggetti del sistema INFEA sono realizzati mediante l'utilizzo del linguaggio Java, principalmente per garantire uno dei più importanti principi guida nella progettazione di INFEA, ossia l'indipendenza dalle caratteristiche hardware e software della piattaforma.

La tecnologia per la programmazione distribuita adottata in INFEA è RMI, che consente di eliminare alcuni limiti imposti dal protocollo HTTP, ad esempio la trasmissione unidirezionale o la mancanza di stato.

Il sistema ha un'architettura client/server a più livelli in cui il primo livello consiste nell'interfaccia utente (*presentation logic*), il secondo comprende la logica dell'applicazione (*business logic*) ed infine il terzo è costituito dal gestore dei dati, un DBMS.

La presentation logic è implementata utilizzando il linguaggio Java, il formalismo HTML ed il linguaggio JavaScript, che ha permesso di implementare anche la parte della business logic sul lato client.

La business logic ed il data access sono implementati tramite applicazioni scritte in Java, mentre i dati sono memorizzati in un DBMS relazionale.

2. ANALISI, STUDIO E PROGETTAZIONE DELL'INTERFACCIA UTENTE

2.1 DISEGNO DELLA NUOVA INTERFACCIA DEL SISTEMA INFEA

Il disegno definitivo dell'interfaccia realizzata è il risultato dell'applicazione dei principi e dei criteri di accessibilità si siti web, allo specifico caso del sistema informativo INFEA; malgrado siano qui esposte le scelte progettuali in accordo solamente alle principali indicazioni dettate dal W3C, durante la fase realizzativa si è provveduto ad interpretare le restanti linee guida.

Ritenendo valido lo schema implementativo dell'attuale interfaccia dal punto di vista architetturale e contenutistico, si è passati ad uno studio che ne valutasse l'usabilità e l'accessibilità secondo le norme apprese e riconosciute.

Il problema principale, riscontrato nell'utilizzo dell'attuale interfaccia basata su browser web da parte di utenti disabili, è quello di non fornire dei chiari punti di riferimento durante l'operazione di ricerca; questo è dovuto da un lato all'ampio utilizzo di frames, dall'altro al fatto che questi frames si sovrappongono fra loro contribuendo, ad una sorta di disorientamento, malgrado forniscano la possibilità di tornare alla visualizzazione del documento precedente.

Per prima cosa si è pensato a come eliminare questo sovrapporsi di contenuti, mantenendo inalterate sia le funzionalità che la quantità di informazioni rese disponibili.

Per migliorare l'accessibilità ma anche l'usabilità dell'interazione tra l'utente ed il sistema, la soluzione pensata è quella di ridurre a 4 il numero di frames, di cui solamente due veramente importanti ai fini della ricerca; gli altri, i cui contenuti si limitano ad abbellimenti grafici, potranno essere lasciati invariati, fatta eccezione per l'eliminazione di tutte le funzionalità fornite con l'impiego di JavaScript e per l'aggiunta dell'attributo ALT ad ogni immagine (linea guida 1 WCAG 1.0).

Si è ipotizzato dunque che l'operazione di ricerca possa essere visualizzata sui due frames più importanti. Il frame di sinistra conterrà tutti i possibili filtri da impostare, e rimarrà fisso senza che vi si sovrappongano ulteriori informazioni; il frame di destra invece mostrerà l'esito delle ricerche e a questo si potranno sovrapporre, a seguito di un'eventuale richiesta dell'utente, i dettagli (informazioni aggiuntive disponibili per ogni Ente) dei risultati ottenuti. Sarà sempre garantita la possibilità di ritornare dal dettaglio di un Ente all'elenco dei risultati, frutto della ricerca.

Partendo dal presupposto di dover eliminare ogni JavaScript presente, si è pensato di inserire tutti i filtri di ricerca all'interno di un unico elemento FORM e di procedere a sequenzializzare tutte le categorie di filtri. A questo proposito, per far sì che non sia necessario scorrere un numero imprecisato di categorie di condizioni prima di raggiungere quella desiderata, si è pensato all'impiego di una tabella come barra di navigazione (linea guida 13); questa tabella conterrà le categorie di filtri selezionabili, tramite la quale raggiungere quella o quelle desiderate, grazie all'uso di ancore per navigare all'interno dello stesso frame.

Una volta avviata la ricerca, i risultati dovranno essere disponibili nel frame di destra e rimanere fissi fintanto che l'utente non decida o di effettuare una nuova ricerca, e in questo caso i nuovi risultati si sovrapporranno a quelli precedenti, oppure di conoscere il dettaglio di un risultato.

Per consentire all'utente di verificare le condizioni impostate, si è pensato di aggiungere nel frame di destra un pulsante che conduca al riepilogo delle condizioni impostate, implementato in due modi: se vi sono filtri selezionati, utilizzando una tabella simile a quella di navigazione presente nel frame di sinistra; oppure, nel caso in cui l'utente non abbia selezionato alcun filtro, visualizzando un messaggio che esprima la mancanza di condizioni impostate. Ai fini progettuali il primo caso è quello più interessante.

La tabella di riepilogo non conterrà tutte le categorie di condizioni, come accade per la tabella di navigazione, ma soltanto quelle per cui l'utente abbia selezionato almeno un filtro, mentre manterrà vuote le

celle corrispondenti a quelle categorie per cui non sia stato selezionato alcun filtro.

Quindi, mentre la tabella presente nel frame di sinistra sarà utilizzata per la navigazione all'interno del documento contenente i filtri (i contenuti delle celle saranno link), quella del frame di destra avrà un ruolo solamente informativo (i contenuti delle celle saranno testo puro) e comparirà, solamente in seguito all'uso del pulsante di riepilogo, allineata alla tabella di sinistra.

L'allineamento orizzontale di queste due tabelle permetterebbe di percepire visivamente quali categorie di condizioni siano state selezionate. Inoltre, questo allineamento consentirebbe all'utente di modificare i filtri impostati, semplicemente utilizzando la solita tabella di navigazione: anche senza specificare il particolare filtro selezionato, sarebbe sufficiente spostarsi sulla tabella di navigazione, cliccare sulle categorie che sono contenute nella tabella delle condizioni selezionate e conoscere specificatamente i filtri impostati.

Si è preferito utilizzare un'unica tabella di navigazione, piuttosto che produrre un inevitabile disorientamento a causa della compresenza di due meccanismi di navigazione molto simili.

Si è inoltre pensato di aggiungere a queste due tabelle una descrizione testuale alternativa – attributo SUMMARY di HTML –, per eliminare quelle difficoltà introdotte dall'uso di alcuni lettori di schermo che non sempre riescono ad interpretare correttamente le righe e le colonne.

Un'altra funzionalità che nell'attuale interfaccia viene implementata attraverso il formalismo JavaScript è quella del conteggio del numero di pagine risultato di una ricerca; il documento contenente l'elenco degli Enti dovrà provvedere ad implementare questa funzionalità senza l'impiego di codice JavaScript.

L'idea di fondo che ha successivamente guidato la realizzazione è stata ripresa dal meccanismo di conteggio dei risultati della maggior parte dei motori di ricerca; al termine di ogni pagina risultato dovrà comparire uno specchietto contenente i collegamenti verso le altre pagine risultato (precedenti e successive) in modo da spostarsi liberamente da un elenco ad un altro.

Visualizzati i risultati di una ricerca, l'utente sarà in grado, cliccando su un particolare Ente, di conoscerne il relativo dettaglio; le informazioni contenute nel dettaglio di un generico Ente saranno raggruppabili in un massimo di 6 sezioni, non sempre tutte presenti.

Il documento contenente il dettaglio si sovrapporrà all'elenco dei risultati e comparirà una lista per favorire la navigazione all'interno del dettaglio, che guiderà l'utente, tramite l'impiego di ancore, alla particolare sezione di interesse.

Questa lista svolgerebbe in questo contesto la medesima funzione che la tabella di navigazione avrebbe nel documento contenente i filtri (linea guida 13). Tuttavia, per non perdere l'esito di una ricerca o semplicemente per consultare il dettaglio di un altro Ente, si è ipotizzata la presenza di un link che da un generico dettaglio riconduca, rimpiazzandolo, all'elenco dei risultati.

In questo modo si garantirebbero facili spostamenti tra i due documenti, risultato e dettaglio.

Grazie a questi accorgimenti la sovrapposizione di frames resterebbe limitata e confinata nel solo frame di destra, senza per questo dover rinunciare alla gran quantità di informazione reperibile con l'attuale interfaccia.

3. REALIZZAZIONE DELL'INTERFACCIA

La progettazione della nuova interfaccia basata sul rispetto delle norme di accessibilità e usabilità web si è realizzata attraverso l'intervento, la modifica o il completo rinnovamento di tre componenti dell'interfaccia attualmente in servizio:

- modifica dei file HTML;
- realizzazione di nuove Servlet Java;
- modifica dei fogli di stile XSL.

Malgrado le varie tappe della realizzazione non siano state così sequenziali e distinte, lo schema sopra elencato descrive in modo consistente e coerente i passi seguiti durante tutta la fase di sviluppo.

3.1 MODIFICA DEI FILE HTML

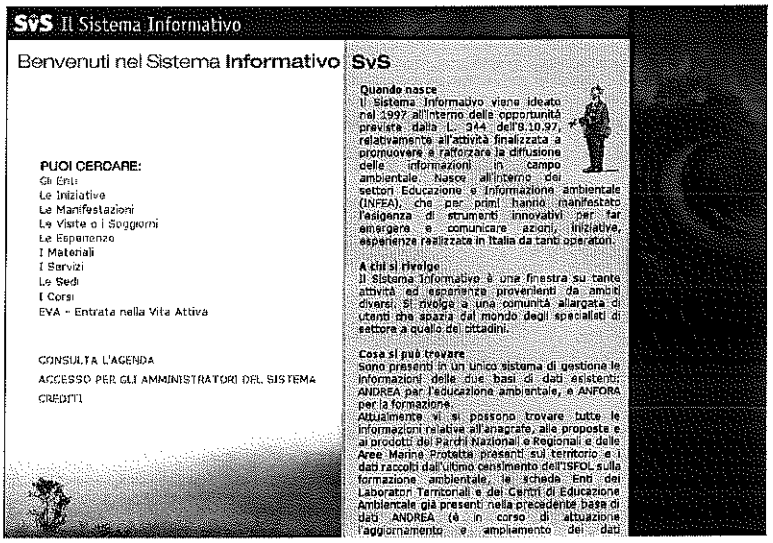
Come tappa iniziale nel processo di sviluppo del prototipo, si è scelto di intervenire sui file HTML, in particolare perché questo avrebbe richiesto uno sforzo minore e avrebbe garantito immediatamente un feedback utile per il proseguimento del lavoro.

Gli interventi su 7 file HTML – la home page e i file impiegati per le operazioni di ricerca degli Enti – si sono basati completamente sulle linee guida WCAG 1.0 del W3C nell'ambito WAI.

Di seguito è descritto il processo di modifica dei file HTML interessati e sono riportati i cambiamenti più rilevanti per la risoluzione dei problemi di accessibilità.

3.1.1 Modifica della home page

A causa della scarsa importanza della home page nel funzionamento del sistema e del suo valore prettamente introduttivo all'interno di INFEA, l'intervento si è limitato a due soli elementi: le immagini e i collegamenti ipertestuali.



- FIGURA 4.1 -

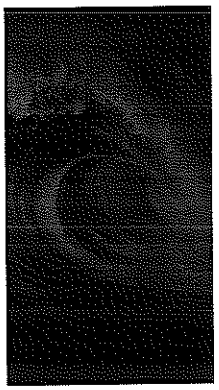
Per quanto riguarda le immagini, si è seguita la linea guida 1, ovvero si è provveduto a fornire un'alternativa testuale equivalente al contenuto visivo dell'immagine, includendo l'attributo ALT all'interno di ogni elemento IMG: il valore dell'attributo ALT rappresenta la descrizione dell'immagine in termini semplici ma altrettanto chiari. In particolare, tale valore distingue l'immagine con un ruolo funzionale e informativo da quella a semplice significato decorativo.

Il seguente esempio mostra l'alternativa testuale scelta per descrivere un'immagine con un ruolo informativo:

PUOI CERCARE:

<IMG src="immagini/iconeprimapagina/h4.gif" width="123" height="15" **ALT="PUOI CERCARE:"**>

Questo esempio mostra, invece, l'alternativa testuale scelta per descrivere un'immagine decorativa dello sfondo:



<IMG height="450" src="immagini/iconeprimapagina/frame_02.jpg" width="279" **ALT="Immagine di sfondo: onda del mare"**>

Mentre nel primo caso il valore dell'attributo ALT coincide con il contenuto testuale dell'immagine, nel secondo si è provveduto a dare una descrizione precisa dell'immagine di sfondo.

L'ulteriore modifica ha interessato i link presenti all'interno della pagina iniziale; tali link sono essenzialmente di due tipi: il primo è costituito da un unico collegamento verso il sito web del Ministero dell'Ambiente, il secondo contiene più link ognuno dei quali conduce ad un altro documento HTML relativo ad una particolare categoria di ricerca (Enti, Servizi, Manifestazioni, ecc...).

Seguendo la linea guida 13 si è cercato di fornire chiari meccanismi di navigazione, aggiungendo ad ogni collegamento ipertestuale l'attributo TITLE che descrivesse la destinazione del link.

Il primo esempio mostra la modifica eseguita sul link diretto verso il sito web del ministero; questo link è rappresentato da una mappa creata su una particolare immagine:

```
<map name="SVS">
        <area shape="rect" coords="3,2 59,33"
href="HTTP://www.minambiente.it/SVS/" target="_top"
TITLE="Collegamento al sito del Ministero dell'Ambiente"></a>
</map>
```

Il secondo esempio riguarda la modifica del collegamento che dirige verso una particolare categoria di ricerca:

```
<a href="ricercaenti/ricercaentiAcc.html" TITLE="Gli Enti">Gli Enti</a>
```

3.1.2 Modifica dei file associati alla ricerca degli Enti

I file associati alla funzionalità di ricerca degli Enti comprendono:

- filtriAcc.html;
- altoentiAcc.html;
- basso_entiAcc.html;
- bludxAcc.html;
- descrizione_enteAcc.html;
- ricercaentiAcc.html

filtriAcc.html

Ai fini della vera e propria ricerca, il file di principale interesse è *filtriAcc.html*, ovvero il frame fisso che compare alla sinistra del video e che contiene tutte le categorie di condizioni selezionabili dall'utente per avviare una ricerca.

Secondo lo schema di progettazione dell'interfaccia descritto nel capitolo precedente, si è dovuto rendere il modello di selezione delle condizioni più semplifice e sequenziale di quello attuale.

Inizialmente si è provveduto alla cancellazione di tutto il codice JavaScript presente nel file, sostanzialmente per due motivi: in primo luogo perché l'impiego di JavaScript è stato altamente sconsigliato dalle norme WAI; inoltre perché, all'interno del disegno della nuova interfaccia che ci si apprestava a realizzare, sarebbe risultato obsoleto e privo di funzionalità. Infatti la progettazione del prototipo ha previsto l'impiego di un frame fisso contenente i filtri senza che a questo si sovrapponessero ulteriori informazioni, in modo da eliminare la necessità di memorizzare tramite JavaScript i campi selezionati.

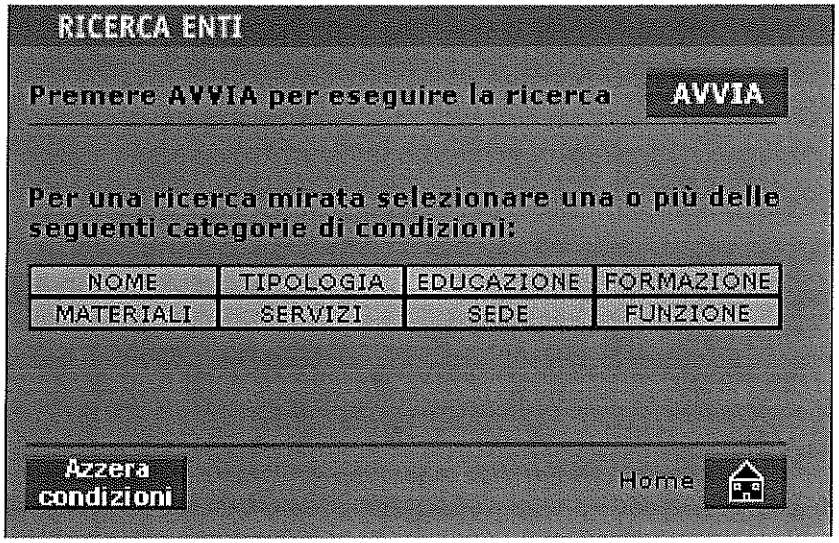
Proprio per questo motivo, i parametri di ricerca impostati dall'utente sono stati inseriti in un unico FORM contenente tutte le condizioni e non in form distinti per ogni categoria di selezione, come accade attualmente.

La formattazione dell'informazione presente in questo documento, come nel caso dell'interfaccia in servizio, si basa sull'impiego di tabelle annidate per l'impaginazione; seppur sconsigliato, questo meccanismo non si è potuto sostituire con altri perché è l'unico a garantire un'impaginazione dell'informazione corretta. Inoltre durante la fase di validazione si è potuto notare che l'uso di tabelle di layout assicurava comunque il rispetto dell'accessibilità delle informazioni.

La funzionalità principale di questo documento è quella di permettere all'utente di avviare una ricerca, restringendo o allargando alcune condizioni attraverso l'impiego di opportuni filtri.

Il documento (figura 4.2) si apre con un'immagine decorativa che descrive la categoria di ricerca a cui si accede tramite la pagina iniziale; tale immagine contiene un'alternativa testuale che ne descrive la

medesima informazione visiva (nel caso del prototipo la categoria di ricerca è rappresentata dagli Enti).



- FIGURA 4.2 -

In seguito a questa immagine introduttiva, viene presentata una mini-guida in forma testuale che descrive brevemente come effettuare una ricerca; in particolare viene spiegato come avviare una ricerca attraverso l'uso del pulsante AVVIA, e in che modo raggiungere le varie condizioni di ricerca tramite l'uso di una tabella di navigazione spiegata in seguito.

Il codice seguente è relativo alla guida per la ricerca:

```
<table width="330" border="0" cellspacing="0" cellpadding="0"
SUMMARY="Tabella per la guida alla ricerca"><!--INIZIO PRIMA TABELLA GUIDA
                                                    ALLA RICERCA-->
    <tr>
    <td height="25" align="left" valign="middle" class="normaleblu"
    width="300">Premere <span class="normaleblu">AVVIA</span> per
    eseguire la ricerca
    </td>
```

```
        <td align="center" valign="middle">
                <INPUT border="0" name="Avvia" type="image"
                src="../immagini/Avvia/AvviaAcc.gif" alt="AVVIA" accesskey="a">
        </td>
        </tr>
        <tr>
        <td colspan="2" height="25" valign="top">
                <img src="../immagini/desktop/riga.gif" width="325" height="1"
                alt="">
        </td>
        </tr>
</table><!--FINE PRIMA TABELLA GUIDA ALLA RICERCA-->

<table width="330" border="0" cellspacing="0" cellpadding="0"><!—INIZIO
                                SECONDA TABELLA GUIDA ALLA RICERCA-->
        <tr>
        <td height="35" align="left" valign="top" class="normaleblu">
        Per una ricerca mirata selezionare una o pi&ugrave; delle seguenti
        categorie di condizioni:
        </td>
        </tr>
</table><!--FINE SECONDA TABELLA GUIDA ALLA RICERCA-->

        <!--SEGUE LA TABELLA DI NAVIGAZIONE-->
```

A differenza di quanto accade nell'interfaccia attuale, le condizioni di selezione sono inserite una di seguito all'altra ma opportunamente distanziate; nel rispetto della linea guida 13, e per impedire che nel passaggio da una condizione ad un'altra si dovesse scorrere l'intero documento, si è cercato di fornire un meccanismo di navigazione all'interno del frame sotto forma tabellare.

Tale tabella, posta nella parte superiore del frame, rappresenta una guida per l'utente: ogni cella contiene il nome della categoria di condizione cui si riferisce.

La tabella è composta da due righe e quattro colonne poiché, nel caso della ricerca di un Ente, otto sono le condizioni di filtri selezionabili;

ogni cella contiene un'ancora all'interno dello stesso documento, espressa dal nome che indica quella determinata condizione.

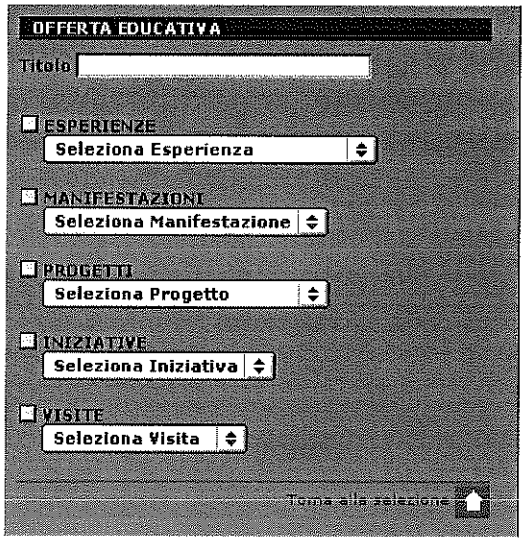
Il codice seguente è relativo alla tabella di navigazione:

```
<table width="328" border="1" cellspacing="0" cellpadding="0"
bgcolor="#C0C8D8" SUMMARY="Tabella per la selezione delle categorie di
condizioni"><!--INIZIO TABELLA FILTRI SELEZIONABILI-->
        <tr>
        <td height="10" width="82" align="center" class="azzurro">
                <A href="#Denominazione" title="Nome">NOME</A>
        </td>
        <td width="82" align="center" class="azzurro">
                <A href="#TipologiaEnte" title="Tipologia">TIPOLOGIA</A>
        </td>
        <td  width="82" align="center" class="azzurro">
        <A href="#OffertaEducativa" title="OffertaEducativa">EDUCAZIONE</A>
        </td>
        <td width="82" align="center" class="azzurro">
        <A href="#OffertaFormativa"
        title="OffertaFormativa">FORMAZIONE</A>
        </td>
        </tr>
        <tr>
        <td  height="10" align="center" class="azzurro">
        <A href="#OffertaMateriali" title="Offerta Materiali">MATERIALI</A>
        </td>

        <td  align="center" class="azzurro">
                <A href="#Servizi"  title="Servizi">SERVIZI</A>
        </td>
        <td  align="center" class="azzurro">
                <A href="#Sede" title="Sede geografica">SEDE</A>
        </td>
        <td align="center" class="azzurro">
                <A href="#Funzione" title="Funzione">FUNZIONE</A>
        </td>
        </tr>
        </table><!--FINE TABELLA FILTRI SELEZIONABILI-->
```

Al di sotto della tabella di navigazione si trovano due pulsanti: il primo serve ad azzerare tutti i filtri impostati fino a quel momento, l'altro rappresenta un collegamento alla pagina iniziale.

Ogni condizione di selezione implementa il medesimo schema di formattazione (figura 4.3): partendo dalla parte superiore, viene caricata un'immagine contenente il nome della condizione e provvista dell'opportuno attributo ALT che ne fornisce una chiara alternativa testuale. Successivamente si possono incontrare diverse tipologie di elementi INPUT – radio button, check box, input testuali – che servono per la selezione effettiva dei campi di ricerca.



- FIGURA 4.3 -

Al termine di ogni condizione è stata inserita un'immagine puramente decorativa (una linea terminatrice) corredata di un'alternativa testuale che specifica il termine della particolare categoria scelta. Al di sotto della linea terminatrice è presente un pulsante, sotto forma di testo e immagine (una freccia rivolta verso l'alto), a cui è associato un link di ritorno all'inizio del frame. Giunto di nuovo alla parte iniziale del frame, l'utente può scegliere o di avviare la ricerca, con le condizioni finora impostate, tramite il pulsante AVVIA, oppure di restringerle ulteriormente o allargarle utilizzando di nuovo la tabella di navigazione per spostarsi all'interno del documento.

Tuttavia, per non costringere l'utente a dover necessariamente ritornare all'inizio del frame nel caso si trovasse alla sezione dedicata all'ultima condizione e volesse avviare subito la ricerca, si è pensato di inserire un secondo pulsante di AVVIA immediatamente sotto alla linea terminatrice.

Questo esempio mostra una parte del codice HTML per la condizione Educazione:

```
<A name="OffertaEducativa"></A><!--ANCORA OFFERTA-EDUCATIVA-->
<table width="350" border="0" cellspacing="0" cellpadding="8"
summary="categoria educazione: prima viene indicato il checkbox e dopo
la specifica condizione a cui il checkbox si riferisce"><!--INIZIO TABELLA
                                                        ESTERNA-->
    <tr>
    <td>
    <table width="330" border="0" cellspacing="0" cellpadding="0">
    <tr>
    <td valign="bottom" height="55">
    <h1><img src="../immagini/BarreColore/OffEdAcc.gif"
    height="16"    width="330" alt="Offerta Educativa">
    </h1>
    </td>
    </tr>
    </table>
    <table width="330" border="0" cellspacing="0" cellpadding="0">
    <tr>
    <td align="left" valign="middle" class="normaleblu" width="40"
    height="35">Titolo</td>
    <td class="normale" align="left" valign="middle">
    <INPUT class="normale" size="25" name="TitoloOffEducativa"
    type="text">
    </td>
    </tr>
    </table>
    <table width="330" border="0" cellspacing="0"
    cellpadding="0"><!--INIZIO TABELLA FILTRI
                                OFFERTA_EDUCATIVA-->
    <tr>
```

```
<td valign="bottom" align="center">
<INPUT type="checkbox" name="TipologiaEducativa_1"
value="1">
</td>
<td height="30" valign="bottom">
<span class="normaleblu">ESPERIENZE</span>
</td>
</tr>
<tr>
<td>
</td>
<td  class="normale" align="left" valign="bottom" width="323">
<span class="normale">
<select class="normale" name="SpecificaEsperienza">
<option selected value="">Seleziona Esperienza</option>
<option value="1">Campo scuola</option>
<option value="2">Citt&agrave; dei bambini</option>
<option value="3">Escursione</option>
...
...
...
</select></span>
</td>
</tr>
... ... ...
... ... ...
... ... ...
<tr>
<td colspan="2" height="20" valign="bottom">
<img src="../immagini/desktop/riga.gif" width="330" height="2"
valign="bottom" ALT="termine categoria Offerta
Educativa">
</td>
</tr>
<tr>
<td colspan="2" class="azzurro" align="right" height="30"
valign="top" >
<A href="#Selezione" TITLE="Torna alla selezione delle
condizioni">
```

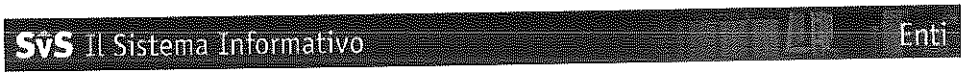
```
Torna alla selezione
<img src="../immagini/FrecciaSuBold.gif" height="24" width="24"
border="0"  align="absmiddle" ALT="(freccia in alto)">
</A>
</td>
</tr>
</table><!--FINE TABELLA OFFERTA_EDUCATIVA -->
</td>
</tr>
</table><!--FINE TABELLA FILTRI ESTERNA-->
```

In questo modo, in accordo con la progettazione, si permette all'utente di spostarsi liberamente da una sezione all'altra del frame tramite la tabella senza dover scorrere il documento fino al punto interessato, mantenendo inalterate tutte le funzionalità attuali e garantendo allo stesso tempo un grado più alto di accessibilità ed usabilità.

Gli altri file HTML, a causa del loro ruolo generalmente decorativo, hanno richiesto una modifica meno incisiva caratterizzata soltanto dal riarrangiamento di alcuni particolari.

altoenti.html

Questo file, che nell'attuale interfaccia ha un ruolo molto importante, nel nuovo prototipo ha una funzionalità prettamente decorativa; contiene un'immagine, a cui è stata aggiunta un'alternativa testuale, sulla quale è stata creata una mappa sensibile per rappresentare un link verso il sito web del Ministero dell'Ambiente:




```
<img src="../immagini/TobBar/TopEnti.gif" width="746" height="37"
usemap="#SVS" border="0" alt="Il Sistema Informativo: categoria
Enti">
<map name="SVS">
        <area shape="rect" coords="3,2 59,33"
        href=HTTP://www.minambiente.it/SVS/
```

```
        title="Collegamento al sito del Ministero dell'Ambiente"
        target="_top">
        </a>
</map>
```

basso_entiAcc.html

Analogamente a quanto detto per il file altoenti.html, anche questo file ha un ruolo puramente grafico; l'immagine che contiene è stata corredata del noto attributo ALT come segue:

```
<img src="../immagini/BottomBar/BottomBarAcc.gif" width="746"
height="28" alt="sfondo blu">
```

bludxAcc.html

Questo file contiene solamente l'immagine principale di sfondo:

```
<img src="../immagini/onda.gif" width="279" height="450"
alt="Immagine di sfondo onda del mare">
```

descrizione_enteAcc.html

Il file di descrizione della categoria Enti viene presentato sul frame di destra (quello su cui verranno visualizzati successivamente i risultati delle ricerche e i dettagli). E' rimasto sostanzialmente inalterato rispetto a quello attuale, ad eccezione dell'inserimento di attributi ALT all'interno degli elementi IMG:

```
<img src="../immagini/Barravuotadx.gif" width="339" height="18"
alt="Inizio della descrizione">
```

ricercaentiAcc.html

Questo file descrive l'insieme dei frame che devono essere caricati una volta entrati nella categoria Enti; è questo, infatti, il file a cui si riferisce il collegamento presente nella pagina iniziale per questa categoria.

In accordo con la linea guida 1, si è cercato di fornire una descrizione chiara di ogni frame; in tal modo sia i browser non grafici, sia i lettori di schermo facilitano all'utente la comprensione del contenuto di un particolare frame.

Di seguito è rappresentata la dichiarazione per il frame relativo all'impostazione dei filtri. L'elemento FRAME è stato corredato di due attributi name e title entrambi con il solito valore: questa scelta dipende dal fatto che spesso vi sono incongruenze tra ciò che viene visualizzato da un browser non grafico e ciò che viene letto da uno screen reader; in questo modo la descrizione del contenuto risulta sempre uguale:

<frame **name="Selezione Condizioni di Ricerca" title="Selezione Condizioni di Ricerca"** src="filtriAcc.html" noresize scrolling="AUTO">

In seguito alle modifiche descritte in questa sezione, si sono aggiunti piccoli interventi resi necessari in seguito ai test di validazione.

3.2 MODIFICA DELLE SERVLET JAVA

I cambiamenti apportati al file HTML contenente le condizioni di selezione si sono riflessi sulla Servlet invocata a seguito di una ricerca (`ElencoEntiServlet`) ed hanno perciò richiesto la realizzazione di una nuova Servlet, costruita ad hoc in base alle modifiche elencate nelle sezioni precedenti (`ElencoEntiAccessibileServlet`).

Un ulteriore intervento, seppur meno deciso, è stato effettuato sulla Servlet che viene invocata a seguito della richiesta del dettaglio di un particolare Ente (`DettaglioEnteServlet`), rinominata `DettaglioEnteAccessibileServlet`.

Vi è però una modifica comune concettualmente ad entrambe le Servlet, che riguarda i fogli di stile XSL da applicare alle risposte in formato XML.

I due fogli di stile XSL che attualmente si occupano di queste trasformazioni sono stati modificati come descritto più avanti; all'interno

di entrambe le Servlet è stato perciò necessario richiamare il foglio di stile opportuno.

In particolare, alla risposta proveniente dalla Servlet `ElencoEntiAccessibile` si è applicato il foglio di stile *ElencoEntiAccessibile.html.xsl*:

```
result=Common.applyXsl(xmlData,Common.schemaDirectory("xsl
schema\\ElencoEntiAccessibile.html.xsl"));
```

mentre per la trasformazione della risposta proveniente dalla Servlet `DettaglioEnteAccessibile` si è applicato il foglio di stile *DettaglioEnteAccessibile.html.xsl*:

```
result=Common.applyXsl(xmlData,Common.schemaDirectory("xsl
schema\\DettaglioEnteAccessibile.html.xsl"));
```

Le sezioni seguenti contengono le descrizioni delle due Servlet, `ElencoEntiAccessibile` e `DettaglioEnteaccessibile`.

3.2.1 Modifica della Servlet per la gestione del risultato di una ricerca

Le funzionalità svolte da questa Servlet sono essenzialmente due:

1. attendere una richiesta da parte di una qualunque tipologia di client (browser web, applet Java...), costruire ed infine inviare al modulo Query Builder una stringa che rappresenti i parametri della richiesta in formato XML;
2. attendere la risposta della query, inviata sempre in formato XML, controllare la tipologia di client che ha eseguito la ricerca ed applicare conseguentemente il foglio di stile XSL opportuno.

Per quanto riguarda la prima funzionalità, a seguito delle modifiche apportate al file filtriAcc.html, la Servlet avrebbe ricevuto i parametri di una ricerca in un modo diverso da quello attuale: l'invio di

un unico elemento FORM presente all'interno del file HTML contenente i filtri, al momento dell'avvio di una ricerca, ha costretto a realizzare una Servlet Java in grado di trattare la ricezione di un singolo FORM contenente tutti i parametri impostati dall'utente, anziché più FORM specifici per ogni condizione.

Al fine di costruire il file XML rappresentante la query con gli opportuni parametri selezionati, la Servlet deve essere prima di tutto in grado di estrarre dal form che riceve i valori di tali parametri.

La Servlet riceve l'intero contenuto del form sotto forma di un oggetto di tipo `HttpServletRequest`:

```
public void doGet(HttpServletRequest req,
HttpServletResponse res)
      throws IOException, ServletException
```

Una volta ricevuta la richiesta, la Servlet invoca un proprio metodo interno (**`buildXmlQuery`**) per poter costruire una stringa rappresentante la query in formato XML.

A questo metodo viene passato come parametro il valore della richiesta, sempre di tipo `HttpServletRequest`:

```
private final String buildXmlQuery(HttpServletRequest req)
```

A questo punto, la nuova Servlet va a estrarre tramite due di questi metodi (**`getParameter`** o **`getParameterValues`**) tutti i valori di ogni condizione, dopodiché esegue alcuni test per verificarne il valore effettivo (per la convenzione usata, il valore è rappresentato da un numero naturale e progressivo nel caso un campo sia settato, oppure **`null`** se il campo è lasciato vuoto).

Di seguito viene riportato il codice per l'estrazione dei valori relativi ai parametri della condizione Educazione:

```
String offertaEASection = "";
String titolo = req.getParameter("TitoloOffEducativa");
      if (titolo != null ) {
            if (!titolo.equals(""))
            offertaEASection+="<TITOLO val='"+titolo+"'/>";
```

```
}
String tipologiaEducativa = "";
String tipologiaEd =
            req.getParameter("TipologiaEducativa_1");
if(tipologiaEd!=null) {
      tipologiaEducativa+="<Esperienza>";
      String specifica =
            req.getParameter("SpecificaEsperienza");
      if (!specifica.equals("")){
            tipologiaEducativa +=
            "<CODSPECIFICATIPOLOGIA val='" +specifica
            + "'/>";
      }
      tipologiaEducativa += "</Esperienza>";
}
else {
      String specifica =
            req.getParameter("SpecificaEsperienza");
      if (!specifica.equals("")){
            tipologiaEducativa += "<Esperienza>";
            tipologiaEducativa +=
            "<CODSPECIFICATIPOLOGIA val='" +
            specifica + "'/>";
            tipologiaEducativa += "</Esperienza>";
       }
}
... ... ... ... ...
... ... ... ... ... //    Vengono  ripetuti  i  controlli
            precedenti per  le altre 4 Tipologie:
... ... ... ... ...  Manifestazioni,  Progetti,  Iniziative  e
            Visite) //
... ... ... ... ...
... ... ... ... ...
}
if (!tipologiaEducativa.equals(""))
      offertaEASection  +=  "<TIPOLOGIA_EDUCATIVA>"  +
      tipologiaEducativa + "</TIPOLOGIA_EDUCATIVA>";
      if(!offertaEASection.equals(""))
            offertaEASection   =   "<OFFERTAEA>"+
            offertaEASection +"</OFFERTAEA>";
```

```
xmlData += offertaEASection; //
```

In base all'esito di questi test, viene costruita dinamicamente una stringa che, restituita dal metodo **`buildXmlQuery`**, rappresenterà la query sotto forma di documento XML; tale stringa sarà inviata al modulo QueryBuilder tramite il metodo **`sendQuery`** presente in una classe apposita (`Common`):

```
String xmlData = (Common.sendQuery(request)).trim();
```

L'oggetto `request` è di tipo `RequestDescriptor`.

Il secondo intervento è stato approntato nell'ambito della seconda funzionalità della Servlet: è stato modificato il file XML contenente i risultati delle ricerche, inviato alla Servlet tramite il modulo Query Builder.

In accordo con il nuovo disegno dell'interfaccia, il frame alla destra del video doveva contenere, oltre ai risultati delle ricerche, anche informazioni relative a quali condizioni erano state impostate (funzionalità implementata nell'attuale interfaccia tramite JavaScript).

In particolare era necessario che il file di risposta XML, prima di essere trasformato tramite XSL, memorizzasse al proprio interno le informazioni relative ai filtri impostati.

Per implementare questa funzionalità si è scelto di creare, all'interno del metodo **`buildXmlQuery`** e parallelamente alla stringa contenente i parametri per la costruzione della query (`xmlData`), un'ulteriore stringa (`xmlFiltri`) che rappresentasse in formato XML i filtri effettivamente impostati.

In base allo schema ideato, la stringa `xmlFiltri` ha la seguente struttura:

```
<FILTRI>
        <FILTRO 1>... ... ...</FILTRO 1>
        <FILTRO 1>... ... ...</FILTRO 1>

            ... ...
            ... ...
                ... ...

    <FILTRO k>... ... ...</FILTRO k>
</FILTRI>
```

Nel caso in cui la ricerca sia avviata senza alcun filtro selezionato – ricerca di tutti gli Enti –, `xmlFiltri` conterrà solamente i tag di apertura e chiusura dell'entità FILTRI.

Una volta creata, la stringa `xmlFiltri` viene opportunamente concatenata con la stringa `xmlData` ottenuta come risposta ad una query (da non confondersi con la stringa `xmlData` per l'invio di una richiesta). Nella gerarchia dell'albero XML di risposta, l'entità FILTRI viene inserita come uno dei nodi figli dell'entità radice ELENCOENTI.

Sarà compito del foglio di stile XSL provvedere a trasformare, adeguatamente alla progettazione, questa informazione aggiuntiva.

Per soddisfare l'ultimo requisito richiesto alla nuova Servlet, è stato realizzato un meccanismo di conteggio delle pagine risultato, che svolgesse lo stesso ruolo di quello attualmente presente implementato con JavaScript.

Per fornire un meccanismo di navigazione all'interno del frame dei risultati, secondo lo schema di progettazione, si è pensato di proporre una soluzione, opportunamente modificata, già adottata da molti motori di ricerca.

Analogamente a quanto accaduto per il problema della memorizzazione dei filtri impostati, è stato necessario memorizzare all'interno del file XML di risposta gli indirizzi corrispondenti a tutti i risultati di ogni ricerca.

A fronte dell'esito di una ricerca, la Servlet tiene conto di tre variabili fondamentali:

- il numero totale dei risultati ottenuti;
- il numero massimo di risultati che si vogliono presentare per ogni pagina (fissato a 10);
- il valore di un parametro che stabilisce la pagina corrente dei risultati.

Ogni volta che ci si sposta in avanti o indietro rispetto alla pagina corrente, non si fa altro che invocare la medesima Servlet con tutti i parametri inalterati, ma con il parametro PaginaCorrente settato con il valore corrispondente alla pagina risultato richiesta.

Ad esempio, supponiamo che la risposta ad una query ci indirizzi verso la seguente Servlet che rappresenta la prima pagina risultato e che il numero totale di pagine risultato sia ***n***:

http://146.48.83.228/Infea2001/ElencoEntiAccessibile?**PaginaCorrente=0**&Denominazione=&TipologiaEnte=0&SpecificaEnte_1=&SpecificaEnte_2=&SpecificaEnte_3=&SpecificaEnte_4=&SpecificaEnte_5=&SpecificaEnte_7=&SpecificaEnte_8=&SpecificaEnte_9=&SpecificaEnte_10=&SpecificaEnte_13=&TitoloOffEducativa=&SpecificaEsperienza=&SpecificaManifestazione=&SpecificaProgetti=&SpecificaIniziativa=&SpecificaVisite=&TitoloOffMateriali=&TitoloOffFormativa=&Area=&Regione=&SiglaProvincia=&Citta=&Avvia.x=20&Avvia.y=4

Per visualizzare una qualsiasi pagina risultato dell'insieme {2,...,n}, basterà modificare l'indirizzo precedente in uno analogo, che abbia come valore del parametro PaginaCorrente uno qualsiasi appartenente all'insieme {2,...,n} e quindi, di fatto, invocare la stessa Servlet con un unico parametro modificato.

Per ogni interrogazione effettuata verso la Servlet, è possibile trovarsi in una delle seguenti situazioni:

a. la pagina dei risultati richiesta è la prima o una qualunque successiva alla prima ed il numero totale di pagine risultato è minore o uguale a 10;

b. la pagina dei risultati richiesta è la prima o una delle successive 9 ed il numero totale di pagine risultato è strettamente maggiore di 10;
c. la pagina dei risultati richiesta è una qualsiasi tra la numero 11 ed il numero totale di pagine risultato.

caso a.

Per ogni pagina risultato, il frame contenente l'esito della ricerca dovrà presentare al suo interno tanti collegamenti rappresentati da numeri naturali, ognuno corrispondente all'indirizzo della Servlet con il valore del parametro PaginaCorrente analogo al numero che lo descrive.

caso b.

La situazione si presenta analoga a quella del caso a., ad eccezione del fatto che è necessario aggiungere un ulteriore collegamento verso le successive 10 pagine risultato.

caso c.

Per tutte quelle pagine risultato che appartengono all'insieme {11,...,n} dove n è il numero totale di pagine, è sicuramente necessario aggiungere a ciò che si è descritto nel caso a. un collegamento verso le 10 pagine precedenti e, nel caso in cui n mod 10 = 1 occorre aggiungere anche un collegamento verso le 10 pagine successive.

Infine, le pagine che cadono nell'intervallo {(n-(n mod 10))+1,...,n} presenteranno solamente un link verso le 10 pagine precedenti.

Per implementare questo meccanismo di conteggio, si è pensato di aggiungere al file XML di risposta una stringa `xmlPage` che descrivesse un'entità particolare, contenente al proprio interno informazioni relative ai collegamenti con le pagine risultato.

Ogni nuova entità nominata `_PAGE` si presenta nel modo seguente:

<_PAGE **val**="12" **totPag**="127"
queryString="PaginaCorrente=12&Denominazione=&Tipologia
Ente=0&SpecificaEnte_1=&SpecificaEnte_2=&SpecificaEnte
_3=&SpecificaEnte_4=&SpecificaEnte_5=&SpecificaEnte_7
=&SpecificaEnte_8=&SpecificaEnte_9=&SpecificaEnte_10=
&SpecificaEnte_13=&TitoloOffEducativa=&SpecificaEsperie
nza=&SpecificaManifestazione=&SpecificaProgetti=&Specifi
caIniziativa=&SpecificaVisite=&TitoloOffMateriali=&TitoloOf
fFormativa=&Area=&Regione=&SiglaProvincia=&Citta
=&Avvia.x=18&Avvia.y=17/>

La nuova entità contiene al proprio interno tre attributi:

- **val** → contiene il numero della pagina risultato;
- **totPag** → contiene il numero totale delle pagine risultato;
- **queryString** → memorizza in una stringa i parametri della query con il valore di PaginaCorrente uguale a quello dell'attributo val.

Ogni file XML di risposta è arricchito con queste nuove entità in modo diverso a seconda dei tre casi sopra descritti:

caso a.

Per ogni pagina risultato vengono create tante entità _PAGE quanto il numero totale di pagine risultato;

caso b.

Per ogni pagina risultato vi sono 10 entità _PAGE e un'entità PAGE_SUCC (rappresentata dalla stringa `xmlPageSucc`), che contiene un attributo nextPage il cui valore è composto dalla stringa completa dei parametri della query, con il valore di PaginaCorrente impostato a 11;

caso c.

Siano n le pagine risultato: come nel caso precedente vi sono 10 entità _PAGE, ad eccezione delle pagine che appartengono all'intervallo {(n-(n mod 10))+1,...,n} per cui ve ne sono n mod 10;

inoltre è presente un'ulteriore entità chiamata PAGE_PREC (contenuta nella stringa `xmlPagePrec`), dotata di un attributo prevPage.

Supponendo di aver richiesto la pagina k tale che 11<=k<=n, il valore di prevPage è costituito dalla stringa completa dei parametri della query nella quale PaginaCorrente vale k-(k mod 10).

Di seguito viene presentato il codice per la gestione del conteggio delle pagine nel caso in cui il file XML di risposta rappresenti l'esito di una query successiva alla prima:

```
String query = req.getQueryString();
if((xmlData.indexOf("_ENTE"))!= -1){ // il file xml di
                                risposta e' successivo al primo

      String prefisso = xmlData.substring(0,pos);
      String suffisso =
      xmlData.substring((pos+1),((xmlData.length())));
      xmlData = prefisso +" "+query_parameters + "
      "+pag_cor +">"+ xmlFiltri;
      int inizio=(((getCurPageNumber(query)-1)/10)*10)+1;
      int fine = ((inizio/10)*10)+10;
      fine = getMin(fine,totPag);
      for(int i=inizio; i<=fine; i++){ // creo tanti
      elementi <_PAGE/> quante sono le pagine risultato;
      ogni entità contiene un campo queryString con i
      parametri della query aggiornati per quella pagina

            xmlPage = "<_PAGE val=\"" +i+ "\" totPag=\""
            +totPag+ "\" queryString=\"" +
            DataUtils.dataEncode(getNextPage(query,i))
            +"\"/>";
            xmlData += xmlPage; // appendo al
                              file xmlData il file xmlPage
      }

            if(((fine-inizio)<(totPag-
            inizio))&&(numero_query!=1)) { // per
            memorizzare l'indirizzo della pagina successiva
```

```
                xmlPageSucc = "<PAGE_SUCC nextPage=\"" +
            DataUtils.dataEncode(getNextPage(query,(fine+1)
            )) +"\"/>";
                xmlData += xmlPageSucc;
            }
            if(inizio>10) { // per memorizzare l'indirizzo
                                della pagina precedente
                 xmlPagePrec = "<PAGE_PREC prevPage=\"" +
      DataUtils.dataEncode(getNextPage(query,(inizio-1)))
      +"\"/>";
                  xmlData += xmlPagePrec;
            }
            xmlData += suffisso;
}
```

I metodi evidenziati in grassetto sono stati impiegati per offrire le seguenti funzionalità:

- **`getQueryString`**: è un metodo predefinito della classe `HttpServletRequest` per ottenere l'intera stringa dei parametri di una query;
- **`getCurPageNumber`**: è un metodo privato realizzato appositamente per catturare il valore, sotto forma di intero, di PaginaCorrente, a cui viene passata come parametro la stringa dei parametri della query:

```
private final int getCurPageNumber(String QUERY) {
      String stringQ = QUERY;
      int position = stringQ.indexOf("&",10);
      String curPage = stringQ.substring(0,position);
      StringTokenizer curPageTok = new
                      StringTokenizer(curPage,"=");
      curPageTok.nextToken();
      int curPageNumber =
      Integer.parseInt(curPageTok.nextToken());
                      return curPageNumber;
}
```

esiste una variante di questo metodo che si chiama **getCurPage**, che ha la stessa funzionalità ma che restituisce il risultato sotto forma di stringa;

- **getNextPage**: è un metodo realizzato ad hoc per modificare il valore di PaginaCorrente presente nella stringa dei parametri della query; ha due argomenti, uno di tipo stringa, che rappresenta la stringa da modificare, e l'altro di tipo intero, che rappresenta il valore da inserire nel campo PaginaCorrente. Restituisce la stringa così modificata:

```
private final String getNextPage(String QUERY,intVALUE) {
      String stringQuery = QUERY; // Stringa dei parametri
                                                  da aggiornare
      int val = VALUE; // nuovo valore da inserire nel
                                    campo PaginaCorrente
      if(val==1){
            val = 0;
      }
      int position = stringQuery.indexOf("&",10);
      String curPage = stringQuery.substring(0,(position-
      1));
      String suffix =
      stringQuery.substring(position,(stringQuery.length()
      ));
      curPage = "PaginaCorrente="+val;
      stringQuery = curPage + suffix;
      return stringQuery;
} //stringa che contiene PaginaCorrente=VALUE
```

Vi è inoltre un metodo privato, realizzato per catturare il numero totale di pagine risultato (**getTotPage**), che nell'esempio sopra citato non viene utilizzato; questo metodo ha due argomenti, uno di tipo stringa e uno di tipo intero per posizionarsi all'interno della stringa. Restituisce un intero:

```
private final int getTotPage(String XMLData, int position)
{
      String XML = XMLData;
      int pos = position;
```

```
        String prefisso = XML.substring(0,pos);
        String suffisso =
              XML.substring((pos+1),((XML.length())));
        int seek = prefisso.indexOf("totRis=");//cerco il
                              numero totale dei risultati
        String totRis = prefisso.substring(seek,seek+13);
        //catturo il numero...
        StringTokenizer RisTok = new
              StringTokenizer(totRis,"\"");
              RisTok.nextToken(); //mi sposto sul token
                                    successivo al primo
        int totaleRis=Integer.parseInt(RisTok.nextToken());
                                    //estraggo un intero
        int totalePag = (totaleRis)/10;
        //eseguo la divisione intera tra il totale dei
        risultati e il fattore di impaginazione (10)
        if(((totaleRis)%10)!=0){
              totalePag++; //incremento di uno il valore di
                                                    totPag
        }
        return totalePag;
}
```

3.2.2 Modifica della Servlet per il trattamento del dettaglio di un Ente

La Servlet che gestisce i dettagli relativi ad ogni Ente può essere invocata, successivamente ad una ricerca, ogni volta che si desiderino conoscere informazioni più specifiche su un particolare Ente risultato.

Il dettaglio relativo ad un Ente viene mostrato sullo stesso frame al posto dei risultati della ricerca; per non avere la completa perdita dell'informazione relativa all'elenco degli Enti risultato, si è dovuto memorizzare l'indirizzo della Servlet `ElencoEntiAccessibile` (con i parametri opportuni) e trascinarlo fino alla Servlet `DettaglioEnteAccessibile`.

Una volta ottenuta la risposta di un dettaglio dalla Servlet `DettaglioEnteAccessibile`, tramite un opportuno foglio di stile XSL, è stato possibile costruire un link diretto verso il documento relativo all'elenco degli Enti risultato. In questo modo, l'utente è in grado di alternare la visualizzazione di due informazioni sul medesimo frame: l'elenco dei risultati di una ricerca e il dettaglio di un particolare Ente appartenente all'elenco.

La Servlet riceve la risposta in formato XML per un certo dettaglio, in seguito all'invio della richiesta nel modo seguente:

```
String xmlData = Common.sendQuery(request);
```

A questo punto occorre modificare la stringa `xmlData`, aggiungendovi la URL da memorizzare; il procedimento si basa sull'utilizzo di alcuni metodi predefiniti dalla Classe `String` di Java, per concatenare in modo corretto la stringa contenente la URL con la stringa `xmlData`.

Come dimostrato prima, modificare la stringa `xmlData` significa modificare il file XML contenente la risposta; per far sì che questo cambiamento fosse conforme alle regole sintattiche di XML, è stato inserito all'interno dell'entità radice SINGOLOENTE un attributo **elencoUrl** il cui valore rappresenta esattamente l'indirizzo dell'elenco degli Enti:

```
int pos = xmlData.indexOf(">",50); // posizionamento
                                   all'interno di xmlData

String prefisso = xmlData.substring(0,(pos-1));
String suffisso =
xmlData.substring(pos,(xmlData.length()));
String elencoURL = "\"
elencoUrl=\""+DataUtils.dataEncode(elencoServletUrl)+"\"";
          // costruzione del nuovo attributo elencoUrl
xmlData = prefisso + elencoURL + suffisso;
// inserimento dell'attributo creato all'interno della
stringa xmlData
```

3.3 MODIFICA DEI FOGLI DI STILE XSL

I fogli si stile XSL modificati consentono la trasformazione dei documenti di risposta XML, prodotti dalle Servlet Java, in documenti XHTML per la visualizzazione su browser web.

Ai fini della realizzazione del prototipo, è stato necessario aggiornare due file attualmente in uso: *ElencoEnti.html.xsl* e *DettaglioEnte.html.xsl*, nominati *ElencoEntiAccessibile.html.xsl* e *DettaglioEnteAccessibile.html.xsl,* descritte di seguito.

3.3.1 Foglio di stile XSL per la visualizzazione dei risultati della ricerca

In base ai criteri di progettazione, lo schema generale di presentazione dei risultati di una ricerca è stato modificato seguendo il medesimo processo di sequenzializzazione delle informazioni attuato per il file filtriAcc.html.

Una volta terminata una ricerca, il foglio di stile esegue delle trasformazioni particolari a seconda di due variabili:

1) la presenza o meno di risultati
2) la presenza o meno di filtri impostati.

Il controllo 1), eseguito tramite operazioni XPath sui nodi dell'albero XML di risposta, produce due diverse situazioni:

a. nessun risultato ottenuto;
b. presenza di almeno 1 risultato.

Il test che viene effettuato è il seguente:

```
<xsl:if test="not(/ELENCOENTI/_ENTE)">
```

ovvero si testa che non vi siano nodi _ENTE, figli del nodo radice ELENCOENTI; se questo test ha esito positivo, significa che siamo nel caso a. e quindi il risultato della ricerca deve evidenziare che non è stato trovato nessun Ente.

A questo punto, è necessario effettuare il controllo 2) per conoscere quali filtri l'utente abbia impostato, in modo da fargli capire quali condizioni selezionate possano aver causato un esito negativo della ricerca.

A questo proposito possono presentarsi due situazioni:

c. nessun filtro selezionato;
d. almeno un filtro selezionato.

È evidente come il caso c. risulti inconsistente una volta verificatosi il caso a., poiché l'avvio di una ricerca senza filtri fornisce tutti gli Enti che fanno parte del sistema INFEA, quindi almeno un risultato.

Si esegue quindi un test, peraltro superfluo, sulla presenza di nodi FILTRI dopodiché, per ognuno di essi, si applica il template relativo:

```
<xsl:if test="ELENCOENTI/FILTRI">
... ... ... ... ...
... ... ... ... ...
... ... ... ... ...
<xsl:apply-templates select="*/FILTRI"/>
... ... ... ... ...
... ... ... ... ...
</xsl:if>
```

Il template relativo ai nodi FILTRI viene descritto in seguito poiché è analogo anche per il caso b..

Se il test di controllo 1) porta nel caso b., occorre visualizzare i risultati della ricerca; inizialmente si fornisce un'informazione sulla pagina corrente e sul numero totale di pagine risultato:

```
<xsl:if test="(ELENCOENTI/@pag_cor)>'0'">
   <span class="blu10">Pagina </span>
   <span class="normaleblu">
   <xsl:value-of select="ELENCOENTI/@pag_cor"/></span>
            //valore di pagina corrente
   <span class="blu10"> di </span>
   <span class="normaleblu">
   <xsl:value-of select="ELENCOENTI/_PAGE/@totPag"/></span>
            //pagine totali
</xsl:if>
```

Viene creato inoltre un pulsante che guida l'utente al riepilogo delle condizioni impostate; tale pulsante è un link all'interno dello stesso documento (ancora) che effettuerà due trasformazioni diverse, in base all'esito del controllo 2).

Il caso b./c. si limita ad informare l'utente del fatto che la ricerca effettuata non aveva nessun filtro impostato; il caso b./d., analogamente a quanto avveniva nel caso a./d. applica per ogni nodo FILTRI l'opportuno template.

Successivamente al pulsante di riepilogo compare l'elenco degli Enti vero e proprio, ottenuto tramite l'applicazione del seguente template:

```
<xsl:apply-templates select="*/_ENTE"/>
```

Infine per dare l'opportunità di scorrere da una pagina risultato ad un'altra, viene creata una piccola tabella con dei numeri che rappresentano il collegamento alla pagina risultato corrispondente.

Se ***n*** è il numero totale delle pagine risultato e n > 1, la tabella creata contiene ***n*** numeri se n ≤ 10; altrimenti contiene 10 numeri, più uno o entrambi tra questi due collegamenti: verso le 10 pagine successive, verso le 10 pagine precedenti.

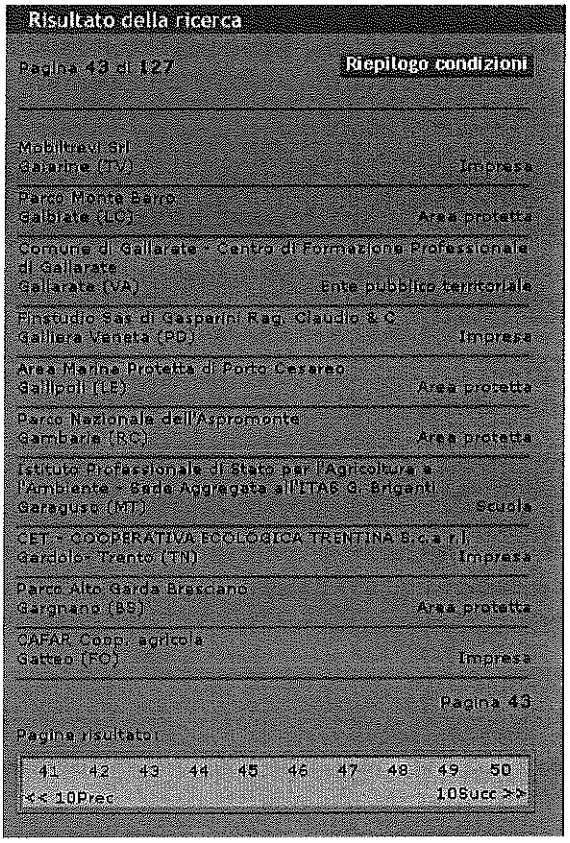
La realizzazione di questo meccanismo, simile ma migliorato rispetto a quelli impiegati dai più comuni motori di ricerca, viene descritta di seguito:

```
<xsl:if test="(ELENCOENTI/_PAGE/@totPag)>'1'">
//guardo se il totale delle pagine è maggiore di 1
        <table border="1"> // creo la tabella cornice
        <tr>
        <td>
                <table border="0" cellspacing="0" cellpadding="0"
                width="322" bgcolor="#C0C8D8"> // tabella
                <tr>
                <xsl:for-each select="ELENCOENTI/_PAGE">
                <xsl:variable name="queryS">
                <xsl:value-of select="@queryString"/>
                //catturo i parametri della query,  attriubuto di _PAGE
                </xsl:variable>
                <xsl:variable name="page">
                <xsl:value-of select="@val"/>
                //catturo il numero di pagina,  attriubuto di _PAGE
                </xsl:variable>
                <xsl:variable name="tot">
                <xsl:value-of select="@totPag"/>
                //catturo il totale delle pagine,  attriubuto di _PAGE
                </xsl:variable>
                <td  align="center" height="15">
<A HREF="http://146.48.83.228/Infea2001/ElencoEntiAccessibile?
{$queryS}" target="Risultati della Ricerca" title="Pagina {$page} di
{$tot}">
                <xsl:value-of select="@val"/>
</A>
        //creo per ogni numero il link corrispondente alla pagina risultato
                </td>
                </xsl:for-each>
                </tr>
                </table>
                <table  border="0"  cellspacing="0"  cellpadding="0"
                width="322" bgcolor="#C0C8D8">
                <tr>
```

```
                    <xsl:if test="ELENCOENTI/PAGE_PREC">
                    // controllo se ci sono nodi PAGE_PREC
                    <xsl:variable name="prevP">
          <xsl:value-of select="ELENCOENTI/PAGE_PREC/@prevPage"/>
                    // catturo l'indirizzo prevPage
                    </xsl:variable>
                    <td  valign="bottom" align="left" height="15">
<A HREF="http://146.48.83.228/Infea2001/ElencoEntiAccessibile?
{$prevP}" target="Risultati della Ricerca" title="10 pagine
precedenti">&lt;&lt; 10Prec
</A>
                    //creo un link verso le 10 pagine precedenti
                    </td>
                    </xsl:if>

                    <xsl:if test="ELENCOENTI/PAGE_SUCC">
                    // controllo se ci sono nodi PAGE_SUCC
                    <xsl:variable name="nextP">
          <xsl:value-of select="ELENCOENTI/PAGE_SUCC/@nextPage"/>
                    // catturo l'indirizzo nextPage
                    </xsl:variable>
                    <td  align="right" height="15">
<A HREF="http://146.48.83.228/Infea2001/ElencoEntiAccessibile?
{$nextP}" target="Risultati della Ricerca" title="10 pagine
successive">10Succ &gt;&gt;
</A>
                    //creo un link verso le 10 pagine successive
                    </td>
                    </xsl:if>
                    </tr>
                    </table>//chiusura della cornice
          </td>
          </tr>
</table>//chiusura della tabella
</xsl:if>
```

La figura 4.4 mostra l'esempio di un possibile esito di una ricerca.



- FIGURA 4.4 -

La descrizione del foglio di stile *ElencoentiAccessibile.html.xsl* risulta più completa con la definizione delle caratteristiche del template dedicato ai nodi FILTRI.

La tabella di seguito mostra i possibili casi finora espressi, in modo schematico; le due righe rappresentano i casi per il controllo 1), mentre le due colonne i casi per il controllo 2):

	c.	d.
a.	Non consistente	- Nessun risultato - Almeno 1 filtro selezionato
b.	- Almeno 1 risultato - Nessun filtro selezionato	- Almeno 1 risultato - Almeno 1 filtro selezionato

In accordo con la tabella, il template per il trattamento dei filtri viene applicato nei casi a./d. e b./d..

La proposta fatta durante la fase di progettazione è stata quella di costruire sul frame dei risultati – frame di destra – una tabella allineata e identica per dimensioni rispetto a quella di navigazione del frame di sinistra.

A differenza di quella di navigazione, questa tabella dovrà contenere solamente le categorie di condizioni per cui è stato selezionato almeno un filtro. In questo modo, nel momento in cui l'utente richiede il riepilogo delle condizioni impostate, se ve ne sono, comparirà una tabella che visivamente risulta allineata con quella di navigazione, ma le cui celle contengono solo le categorie per cui è stato impostato almeno un filtro.

È da notare che la tabella delle condizioni è solamente una tabella informativa e non contiene nessun collegamento verso altre parti del documento; al contrario, la tabella di navigazione rappresenta l'unico meccanismo per potersi agevolmente spostare tra una categoria di filtri e l'altra. In questo modo si evita la confusione ed il disorientamento dovuti alla compresenza di due tabelle di navigazione simili.

Di seguito viene riportata la trasformazione eseguita dal template per FILTRI:

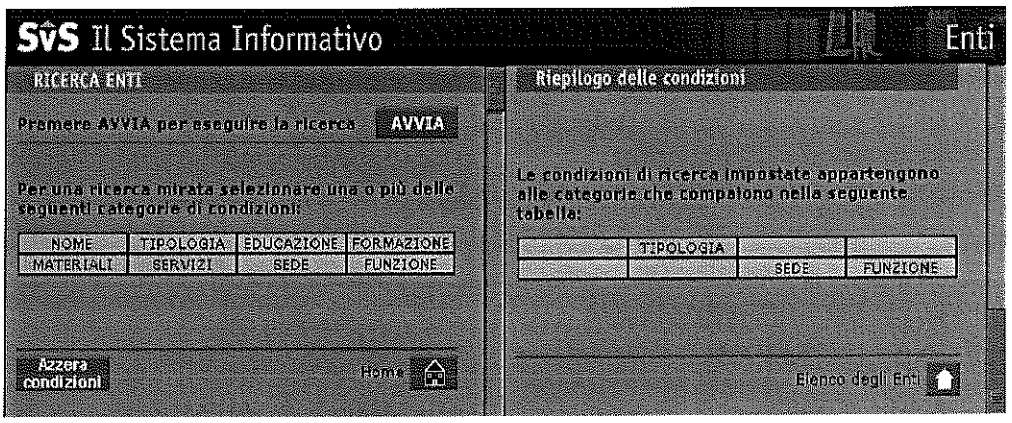
```
<xsl:template match="FILTRI">
<table width="328" border="1" cellspacing="0" cellpadding="0"
bgcolor="#C0C8D8">
//creo la stessa tabella presente nel frame contenente i filtri e la riempo
con i SOLI filtri impostati
        <tr> // prima riga
        <td height="10" width="82" align="center" class="azzurro">
                // prima colonna
        <xsl:choose>
        <xsl:when test="_DATIANAGRAFICI">NOME</xsl:when>
        //controllo se e' stato impostato il nome dell'ente
        <xsl:otherwise>   </xsl:otherwise>
        //altrimenti inserisco uno spazio vuoto
        </xsl:choose>
        </td>
```

```
<td width="82" align="center" class="azzurro">// seconda
                                                colonna
<xsl:choose>
<xsl:when test="_TIPOLOGIA">TIPOLOGIA</xsl:when>
//controllo se e' stata impostata la tipologia dell'ente
<xsl:otherwise>   </xsl:otherwise>
</xsl:choose>
</td>
... ... ... ... ...
... ... ... ... ...
</tr>// fine prima riga
<tr>// inizio seconda riga
<td height="10" align="center" class="azzurro">// prima colonna
<xsl:choose>
<xsl:when test="_MATERIALE">MATERIALI</xsl:when>
//controllo se e' stato impostata un'offerta materiali
<xsl:otherwise>   </xsl:otherwise>
</xsl:choose>
</td>

<td align="center" class="azzurro">// seconda colonna
<xsl:choose>
<xsl:when test="_SERVIZI">SERVIZI</xsl:when>
//controllo se e' stato impostato un servizio
<xsl:otherwise>   </xsl:otherwise>
</xsl:choose>
</td>
... ... ... ... ...
... ... ... ... ...
</tr>// fine seconda riga

</table>// fine tabella
</xsl:template>
```

La figura 4.5 mostra un particolare scenario che si può presentare all'utente a fronte di una richiesta di riepilogo delle condizioni impostate.



- FIGURA 4.5 -

La progettazione della nuova interfaccia ha richiesto qualche altro intervento che in questa sezione non viene descritto perché meno rilevante rispetto a quelli finora evidenziati.

3.3.2 Foglio di stile XSL per la visualizzazione del dettaglio di Ente

Anche il documento relativo al dettaglio dell'Ente è stato notevolmente cambiato grazie al foglio di stile *DettaglioEnteAccessibile.html.xsl*.

Per poter gestire la stessa quantità di informazioni presenti attualmente in un generico dettaglio, si è dovuto rappresentare il dettaglio in due parti visivamente contigue ma distinte: la parte iniziale del documento fornisce i dettagli più comuni e meno specifici dell'Ente scelto, mentre la seconda parte è costituita da una lista di ulteriori

dettagli che viene usata per navigare all'interno delle varie sezioni elencate, tramite il solito meccanismo delle ancore.

Questa lista è creata dinamicamente e specificatamente per un certo Ente, poiché non tutti gli Enti presentano le stesse categorie di dettaglio; per evitare di inserire nella lista categorie che non sono presenti per un dato Ente, si effettuano dei test tramite le espressioni XPath:

```
<xsl:if test=
"(normalize-space(_DATIANAGRAFICI/IDENTESOVRAORDINANTE/@val)
!='')
or (normalize-space(*/IDSEDENAZIONALE/@val) !='') or (_TIPOLOGIA)
or(SITOWEB)">
<tr>
<td align="left" height="30" valign="middle">
<img src="infea-test/accessibile/immagini/SimboloLista.gif" alt=""/>
<A href="#Informazioni" title="Informazioni generali sull' Ente">
<span class="azzurrobig"> INFORMAZIONI GENERALI</span></A>
</td>
</tr>
<tr>
<td>
<img src="infea-test/accessibile/immagini/desktop/riga.gif" width="333"
alt=""/>
</td>
</tr>
</xsl:if>
<xsl:if test="_SERVIZI/_SRVZ">
<tr>
<td align="left" height="30" valign="middle">
<img src="infea-test/accessibile/immagini/SimboloLista.gif" alt=""/>
<A href="#Servizi" title="Servizi">
<span class="azzurrobig"> SERVIZI</span></A>
</td>
</tr>
<tr>
<td>
<img src="infea-test/accessibile/immagini/desktop/riga.gif" width="333"
alt=""/>
```

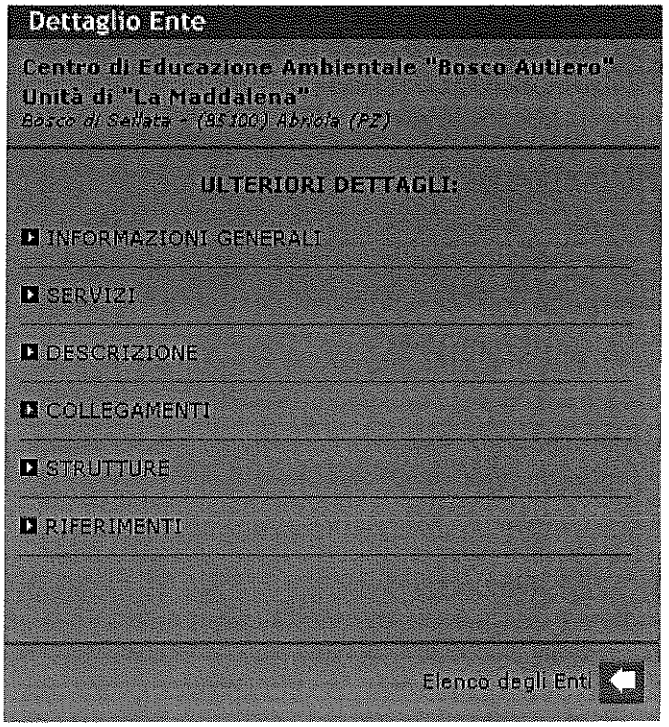
```
</td>
</tr>
</xsl:if>
... ... ... ... ...
... ... ... ... ...
... ... ... ... ...
```

Ogni collegamento è dotato dell'attributo TITLE, mentre l'immagine che rappresenta il simbolo della lista puntata ha l'attributo ALT volutamente vuoto.

Successivamente alla lista di navigazione è presente un pulsante per il ritorno all'elenco dei risultati:

```
<xsl:variable name="ELENCO_URL">
<xsl:value-of select="/SINGOLOENTE/@elencoUrl"/>
//catturo l'indirizzo dell'elenco degli enti
</xsl:variable>
<A HREF="{$ELENCO_URL}" title="Ritorna all'elenco degli Enti"
TARGET="Risultati della Ricerca">
<span class="azzurrobig">Elenco degli Enti </span>
<img src="infea-test/accessibile/immagini/FrecciaBackBold.gif" border="0"
align="absmiddle" alt="freccia indietro"/>
</A>
```

La figura 4.6 mostra la parte iniziale del dettaglio di un particolare Ente; si noti il meccanismo di navigazione interno al documento, implementato tramite una sorta di lista puntata, ed il pulsante per il ritorno all'elenco degli Enti risultato.



- FIGURA 4.6 -

Per poter ritornare alla lista di navigazione all'interno del dettaglio, si è dovuto aggiungere, al termine di ogni sezione, un pulsante che riportasse al menù principale:

```
<A href="#Dettaglio" title="Torna al Dettaglio dell'Ente">
// ancora all'inizio del documento
<span class="azzurrobig">Dettaglio dell'Ente </span>
<img src="infea-test/accessibile/immagini/FrecciaSuBold.gif" border="0"
align="absmiddle" alt="freccia in alto"/>
</A>
```

Entrambi i pulsanti descritti sopra sono costituiti da un collegamento ipertestuale, composto da un'immagine – una freccia rivolta verso sinistra o verso l'alto – e da testo; come sempre, sia l'elemento IMG che l'elemento A contengono rispettivamente gli attributi ALT e TITLE.

Grazie a questi meccanismi, il dettaglio viene visualizzato sullo stesso frame dell'elenco, ma al tempo stesso risulta facile navigare all'interno delle varie sezioni tramite la lista e ritornare alla schermata contenente l'esito della propria ricerca.

4. VALIDAZIONE DEL PROTOTIPO REALIZZATO

L'ultima fase di lavoro è stata completamente dedicata alla validazione del prototipo realizzato; dapprima si sono effettuati dei test che valutassero le funzionalità del sistema attraverso l'impiego di un browser testuale (Lynx 2.8.3 per Windows e Lynx 2.7.1 per Macintosh), successivamente è stata specificatamente valutata l'accessibilità del prototipo tramite l'ausilio di due tools utilizzabili direttamente via web (*Lift* [9] di Usable.net e *Bobby* di Cast).

Inoltre un ricercatore non vedente esperto della problematica, ha redatto un documento che ha evidenziato i difetti da lei riscontrati durante le prove effettuate nell'arco di una settimana.

I risultati di tutti questi test hanno fornito dei validi spunti per effettuare alcuni piccoli ritocchi, che hanno consentito al prototipo realizzato di soddisfare i requisiti di accessibilità ed usabilità richiesti.

4.1 LYNX

Il punto 4) relativo alla fase di validazione raccomandata dal W3C sui temi usabilità e accessibilità si riferisce all'impiego di un browser testuale per testare la visualizzazione di documenti web.

Il browser testuale più diffuso è senza dubbio Lynx; tale browser, scaricabile gratuitamente, rappresenta un valido banco di prova per verificare quanto un'interfaccia progettata e realizzata su un ambiente grafico (MS Explorer o Netscape Navigator) si adatti ad un ambiente completamente testuale.

Lynx è stato scelto per due ragioni principali: innanzitutto rappresenta uno strumento usato tuttora da una grande fetta di utenti disabili; inoltre costituisce il supporto di visualizzazione web più diffuso nei paesi in via di sviluppo.

Lynx può essere usato come strumento cui uno screen reader fa riferimento per la lettura di un documento web; tuttavia il livello tecnologico raggiunto dai lettori di schermo tende ad orientare anche

---

[9] http://www.usablenet.com

l'utente disabile verso l'impiego di browser grafici anziché browser testuali.

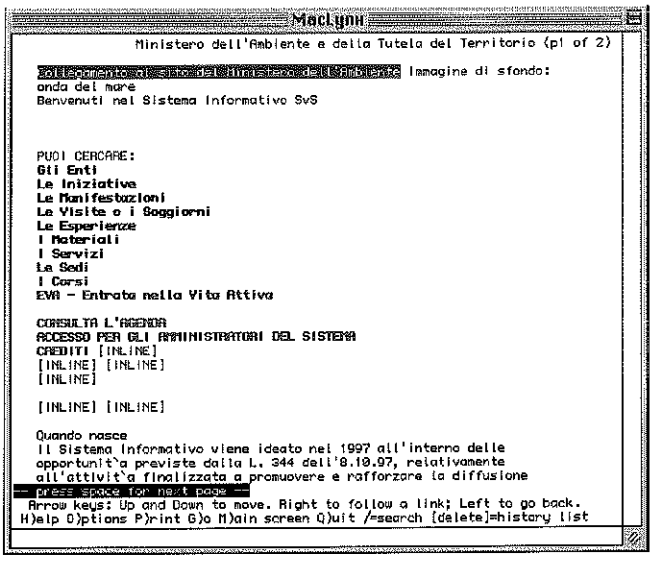
Ciononostante è altrettanto significativo il largo impiego di Lynx come browser per utenti non disabili, specialmente nei paesi in via di sviluppo.

Malgrado le versioni di Lynx utilizzate riconoscano i frame HTML, la presentazione dei contenuti informativi degli stessi frame è resa in modo strettamente sequenziale; in particolare, Lynx indica all'utente la presenza di tutti gli elementi frame, ma è possibile visualizzare il contenuto di un solo frame alla volta.

In questa situazione, la visione parziale che si ottiene necessita di alcuni accorgimenti tecnici per essere adattata alla visione di insieme che si ha con un browser grafico; si tratta di modifiche semplici, il cui obiettivo è sostanzialmente quello di non far perdere l'orientamento all'utente, offrendogli dei nuovi punti di riferimento, come immagini o link visibili soltanto utilizzando un browser testuale e nascosti a qualsiasi browser grafico.

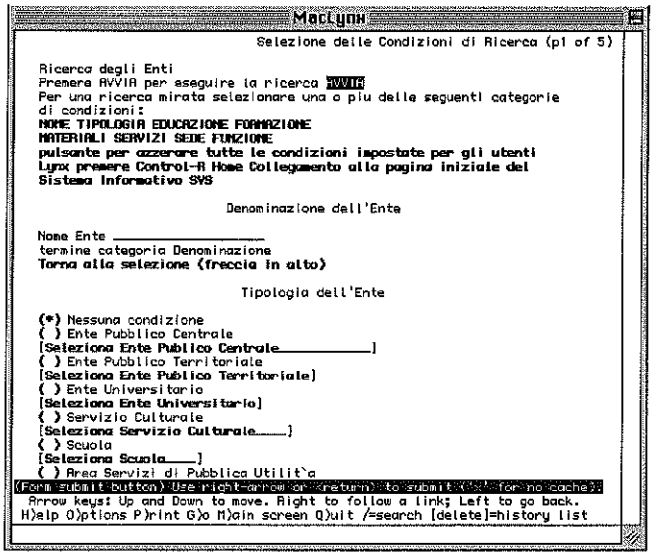
I test effettuati vertevano soprattutto sulle funzionalità dell'interfaccia testuale durante un'operazione di ricerca e la richiesta di un dettaglio.

La figura 5.1 mostra la visualizzazione della home page.



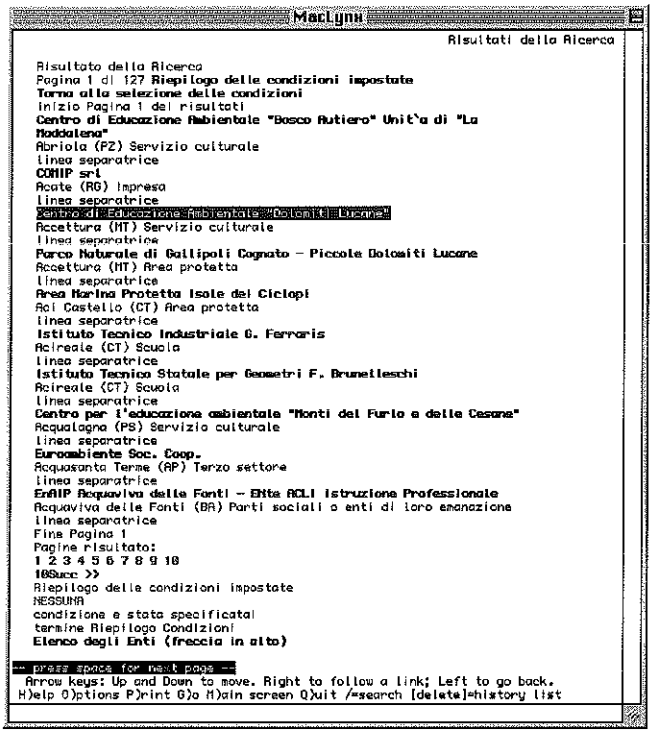
- FIGURA 5.1 -

La figura 5.2 mostra l'interfaccia per la selezione delle condizioni.



- FIGURA 5.2 -

La figura 5.3 mostra il possibile esito di una ricerca.



- FIGURA 5.3 -

4.2 TOOLS AUTOMATICI VIA WEB

Vi sono molte risorse sul web che trattano gli argomenti relativi all'accessibilità e all'usabilità dei documenti, oltre che alla risoluzione tecnica dei problemi che più spesso si incontrano durante la realizzazione di un servizio web.

Uno dei più importanti siti web che rappresenta un esempio di tali risorse è senza dubbio www.webusabile.it; questo sito concentra al proprio interno sia informazioni teoriche riguardanti l'accessibilità e l'usabilità (come ad esempio le linee guida WCAG 1.0), sia strumenti tecnici per la validazione on line di uno o più documenti web.

Il punto di forza di questi strumenti è senz'altro la facilità con la quale possono essere usati: è sufficiente specificare la URL del documento web che si intende validare e l'indirizzo di posta al quale si desidera ricevere l'esito del test. Anche le modalità d'uso risultano particolarmente semplici: si tratta di tools per lo più gratuiti, che non pongono alcun limite relativo alle volte in cui possono essere utilizzati.

In questo ampio panorama di strumenti che il web stesso mette a disposizione dello sviluppatore, si è pensato di citare due importanti validatori automatici: Bobby di CAST e Lift di Usable.net. Entrambi hanno in comune la facilità d'uso, malgrado presentino caratteristiche diverse: da un lato Bobby costituisce probabilmente lo strumento più famoso e più completo per la verifica dell'accessibilità di un documento web; dall'altro Lift rappresenta uno strumento più semplice e pulito che, malgrado offra meno dettagli, consente allo sviluppatore di intervenire più facilmente sui difetti riscontrati.

L'utilizzo di Bobby, infatti, ha supportato una minima parte dell'intera fase di validazione, poiché il livello di dettaglio particolarmente accurato a cui testa i documenti ha prodotto degli esiti difficilmente interpretabili, e ha evidenziato qualche limite alla flessibilità con cui interpretare le informazioni.

Al contrario, Lift è stato più volte usato a fronte delle modifiche che esso stesso proponeva; l'interfaccia utente risulta gradevole e, oltre a evidenziare i difetti all'interno del codice (HTML), suddivide gli errori per ordine di priorità, dal più grave al più veniale, seguendo la medesima impostazione dei livelli di priorità delle linee guida del W3C.

Ad ogni difetto era associato un riferimento all'interno del documento, e una possibile soluzione che ha permesso di correggere errori più rilevanti.

4.3 COLLABORAZIONE CON UN UTENTE DISABILE

L'ultima parte relativa alla fase di validazione, e presumibilmente la più rilevante, è stata effettuata con la collaborazione di una persona non vedente, attualmente impegnata in un dottorato di ricerca presso il gruppo di Human-Computer Interaction dell'ISTI di Pisa.

In accordo con i principi sulle metodolgie di validazione espressi dal W3C, si è pensato di far revisionare il prototipo realizzato, e in parte già validato, ad un utente disabile esperto.

tale utente ha eseguito le prove in maniera autonoma nell'arco di una settimana, in primo luogo per acquisire una certa confidenza con il funzionamento del sistema, e secondariamente perché una semplice revisione congiunta avrebbe evidenziato solo una minima parte degli eventuali malfunzionamenti.

Grazie alla sua collaborazione, è stato possibile conoscere il comportamento dell'interfaccia con l'ausilio di un lettore di schermo (Jaws 4.50 per Windows) su una piattaforma PC Windows ME con MS Internet Explorer (versione 5.0 e 5.5).

Quest'ultima fase di test ha prodotto un documento redatto dalla stessa Barbara Leporini, in cui si evidenziano i principali problemi riscontrati nell'utilizzo del prototipo, divisi in 4 sezioni distinte: *Frame, Tabelle, Button, CheckBox e ComboBox, Intestazioni.*

Ogni malfunzionamento è ampiamente spiegato sia dal punto di vista tecnico, individuando la parte di codice HTML che lo causa, sia dal punto di vista pratico, attraverso dei semplici esempi di scenari poco considerati durante la realizzazione. Al termine di ogni problema indicato, vengono espresse alcune possibili soluzioni per risolverlo.

La stesura di questo documento si è rilevata molto utile e soprattutto attendibile: concentrando l'ultima parte dell'attività di tirocinio sui malfunzionamenti qui riscontrati, è stato possibile apportare

importanti modifiche essenziali al fine della completa realizzazione del prototipo.

CONCLUSIONI

Quando uno sviluppatore si trova a dover progettare e realizzare un'interfaccia uomo-macchina, una delle principali difficoltà che può incontrare è quella di doversi 'immedesimare' nell'utente; e questa difficoltà si fa ancora maggiore nel caso in cui l'utente presenti esigenze diverse da quelle medie, a causa di disabilità fisiche o mentali o per gli strumenti di cui dispone. È allora che sorgono le problematiche della cosiddetta progettazione universale.

Adattare o realizzare ex novo un'interfaccia secondo i criteri di progettazione universale significa dover risolvere una quantità di problemi, che derivano essenzialmente dalla moltitudine di fattori da tenere in considerazione: si pensi banalmente a come i diversi tipi di disabilità (ad esempio visiva o motoria) determinino nell'utente esigenze diverse.

Fortunatamente, l'impegno della tecnologia è sufficientemente bilanciato: da un lato lo sviluppatore possiede strumenti che descrivono nel dettaglio alcuni accorgimenti generali (ad esempio gli standard WAI del W3C); dall'altro l'utente può affidarsi ad un robusto insieme di ausili e meccanismi ottimizzati per molti tipi di disabilità (lettori di schermo, dispositivi di puntamento alternativi al mouse...), che costituiscono le cosiddette tecnologie assistive.

Questo bilanciamento degli sforzi favorisce sia lo sviluppatore, che trova risposte esaurienti e possibili strategie di soluzione, sia l'utente che, grazie alle tecnologie assistive, non deve rinunciare ai benefici dello sviluppo.

Il prototipo dell'interfaccia web per il sistema INFEA è un esempio di come le funzionalità e l'efficienza possano rimanere sostanzialmente inalterate anche a fronte di un ridisegnamento dell'interfaccia.

Realizzare un'interfaccia più usabile ed accessibile di quella attualmente in servizio non ha generato un impoverimento dei contenuti o un annullamento dell'aspetto decorativo; al contrario, mantenendo l'impostazione grafica attuale, è stato possibile costruire un'interfaccia chiara e pulita.

Dato l'ampio spettro di problematiche cui si è dovuto far fronte, la presenza di alcuni limiti è senz'altro ipotizzabile, sebbene non sia stata

riscontrata durante la fase di validazione del prototipo. È comunque bene sottolineare che la fase di test, per quanto robusta, è rimasta tuttavia incompleta secondo le norme del W3C, a causa della ristrettezza dei tempi di realizzazione.

Oggi l'informatico possiede talmente tanti strumenti e conoscenze, da indurre i 'profani' a credere che l'Informatica rappresenti una sorta di passe-partout per risolvere tutti i problemi presenti e futuri del genere umano, ma paradossalmente in grado di produrne altrettanti.

Sebbene questa convinzione possa rivelarsi in parte fondata, è innegabile che l'Informatico possa scegliere ad ogni passo di non aggiungere altri problemi a quelli con cui già convive, possa cercare sempre di aggiustare piuttosto che di distruggere.

La disabilità e l'integrazione dei disabili nella società cilvile non sta tra i primi posti nell'ipotetica graduatoria dei problemi che affliggono il mondo, dove regnano incontrastate le guerre e la povertà; in realtà, non serve poi tanto per accorgersi che dietro all'inserimento sociale dei disabili si nasconde solo un altro capitolo dell'annosa questione della discriminazione.

Così come è discriminante impedire l'uso di un qualsiasi servizio pubblico ad una persona disabile, è altrettanto discriminante concedere solo alle persone cosiddette normali il beneficio che si trae da una nuova tecnologia.

Internet è una nuova tecnologia e il web è un servizio che si basa su questa tecnologia. Impedire, o comunque limitare sensibilmente, l'accesso alle informazioni distribuite sul web è a tutti gli effetti un problema di discriminazione.

Un altro ostacolo da buttare giù. E ci vuole tempo, impegno e costanza.

BIBLIOGRAFIA

[1] **Basi di Dati Relazionali e ad Oggetti**
di Albano A., Ghelli G., Orsini R.
ed. Zanichelli (2000)

[2] **Web Architectures for Database Access**
(Atti del convegno WebNet "World Conference 1999 on the WWW and Internet", Honolulu – Hawaii, USA)
articolo pgg. 1473-1475
di Aloia N., Concordia C, Miori V.

[3] **Web-based Information Systems**
(Atti del convegno "16th World Computer Congress 2000 Information Technology for Business Management", Pechino – Cina)
articolo pgg. 581-588
di Aloia N., Concordia C, Miori V.

[4] **Architettura software, funzionalità e meccanismi del Sistema INFEA**
di Tardelli S., Versienti L.
rapporto CNUCE-B4-2002-004 (2002)

[5] **HTML: The Complete Reference**
di Powell T.A.
ed. McGraw-Hill (2000)

[6] **HTML for the World Wide Web with XHTML and CSS: Visual QuickStart Guide (5th Edition)**
di Castro E.
ed. Peachpit Pressù (2002)

[7] **Java Servlet Programming Bible**
di Rajagopalan S., Rajamani R., Krishnaswamy R., Vijendran S.
ed. John Wiley & Sons (2002)

[8] **Java & XML (2nd Edition) - Solutions to Real-World Problems**
di McLaughlin B.
(2001)

[9] **Java Servlet Programming (2nd Edition)**
di Hunter J., Crawford W.
(2001)

[10] **http://www.latoserver.it/java/servlets**
articolo di Capodieci G. del *24/11/2001*

[11] **XML Step by Step**
di Young M.
ed. Microsoft (2000)

[12] **XML Guida di Riferimento**
di Harold E.R., Means W.S.
ed. O'Reilly (2000)

[13] **XML Bible (2nd Edition)**
di Harold E.R.
ed. John Wiley & Sons (2001)

[14] **I disabili nella società dell'Informazione - Norme e tecnologie**
di AA.VV.
ed. Franco Angeli (2002)

[15] **Considerazioni per la realizzazione di sistemi informativi basati su web accessibili ai disabili**
(6° convegno "Informatica Didattica e Disabilità")
di Aloia N., Concordia C., Furfari F., Miori V.

[16] **http://www.w3.org/TR/WCAG**
http://www.aib.it/aib/cwai/WAI-trad.htm *vers. italiana*

linee guida sull'accessibilità dei contenuti web (WCAG 1.0)

[17] **http://usability.babel.it**
criteri e metodologie di progettazione di interfacce usabili

[18] **http://www.webusabile.it**
criteri, metodologie e supporti di validazione per interfacce web usabili

[19] **http://www.useit.com**
sito ufficiale di Jakob Nielsen