UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Creazione e gestione di EVT multilingua utilizzando le API Storage, all'interno del progetto Vercelli Book digitale.

Candidato: *Luca Sarri*

Relatore: *Roberto Rosselli del Turco*

Correlatore: *Maurizio Tesconi*

Anno Accademico 2014/2015

Indice

Introduzione

Codifica di testi è una disciplina nuova, il cui scopo è rappresentare un testo scritto su supporti cartacei in una versione digitale leggibile da un elaboratore.

Chi si occupa di codifica di testi si affida alle linee guida offerte dalla TEI (Text Encoding Initiative), al fine di ottenere un edizione digitale efficiente che metta a disposizione dell'utente strumenti adatti per lavorare sul testo.

In questo elaborato vedremo le caratteristiche che un edizione digitale deve rispettare per essere valida, in particolare verrà analizzata la digitalizzazione del Vercelli Book e la creazione della sua interfaccia grafica EVT (Edition Visualization Technology) ideata e sviluppata dal Professor Roberto Rosselli Del Turco e da un team di studenti di Informatica Umanistica. EVT fa uso delle ultime tecnologie messe a disposizione per la progettazione web, la mia ricerca analizzerà nel dettaglio HTML5 e in particolare una sua nuova funzionalità: le API Storage, che permettono l'immagazzinamento dei dati direttamente nel browser.

La mia Tesi per il conseguimento del titolo triennale in Informatica Umanistica è strutturata in cinque capitoli: nel primo vedremo cos'è il Vercelli Book, analizzeremo le teorie sul suo arrivo in Italia e sulla sua datazione. Il secondo capitolo introduce la digitalizzazione del testo, quali vantaggi porta e quali sono le caratteristiche che un interfaccia grafica deve rispettare per poter formare una buona edizione digitale. Il terzo capitolo analizza il progetto del Vercelli Book Digitale, mostrando le attuali e le future caratteristiche di EVT, il software che permette la lettura del manoscritto custodito a Vercelli. Nel quarto capitolo si analizza HTML5 quali obbiettivi vuole raggiungere e le sue nuove funzionalità, in particolare analizziamo le API Storage cosa sono e come si utilizzano. Nel quinto e ultimo capitolo prendo in esame il mio progetto, descrivendolo passo passo e integrando la descrizione con parti di codice significativo. Una volta ultimata la spiegazione del codice descrivo come è avvenuta la sua integrazione all'interno di EVT come e quali file ho dovuto modificare per integrare il mio progetto all'interno delle trasformazioni XSLT.

La scelta di chiedere al Professor Del Turco di poter sostenere la tesi con lui è dovuta al fatto che la disciplina della codifica di testi incarna per me l'essenza della facoltà di “Informatica Umanistica”, in quanto è una materia recente come la facoltà e fonde al suo interno materie informatiche come la codifica e la programmazione (le materie che più mi hanno interessato durante il corso di laurea), e materie umanistiche avendo a che fare con testi spesso di grande valore letterario, in questo caso il Vercelli Book.

Capitolo 1

Vercelli Book

Descrizione del codice

Il Vercelli Book conosciuto anche con il nome di Codex Vercellensis è un manoscritto scritto alla fine del X secolo e conservato oggi a Vercelli presso il Palazzo Arcivescovile nella Biblioteca Capitolare sotto la segnatura di Codex VII. È una collezione di opere scritta in inglese antico ed è formata da un insieme di scritti a carattere religioso in prosa e in versi.

Il manoscritto fu redatto dal committente per riunire testi in modo da costruirsi un florilegio spirituale, utile per la meditazione e la preghiera durante il suo pellegrinaggio; è dunque una collezione originale messa insieme quasi sicuramente in area kentica[1] da un'ampia varietà di scritti omiletici o salvifici di autori di età diversa scelti fra quelli disponibili nel sud dell'Inghilterra e copiati durante un lungo periodo di tempo.

Non sappiamo quale biblioteca episcopale o monastica contenesse una così vasta raccolta di materiale, tra le più probabili la biblioteca di Rochester o quella del monastero di Sant'Agostino a Canterbury.

Il Vercelli Book contiene 23 omelie in prosa di grande interesse linguistico e culturale (11 delle quali attestate unicamente in questo codice) e 6 componimenti poetici, è composto da 136 fogli di sottile pergamena delle dimensioni di circa 31x20cm ognuno dei quali contiene dalle 23 alle 32 righe di testo scritte in minuscola quadrata anglosassone[2] con alcune iniziali in forma zoomorfica elaborata[3].

Come tutti i libri medioevali il codice è un unità costruita da una serie di unità più piccole assemblate in successione; la sua struttura base è difatti il fascicolo, non la pagina singola, che consisteva nell'unione di diversi fogli incorporati gli uni negli altri, solidali due a due, in quanto provenienti da un foglio piegato a

---

1 Area situata a nordovest di Londra, lungo il fiume Fleet, oggi sotterraneo.

2 Cfr. Vercelli Book Digitale http://vbd.humnet.unipi.it/?page_id=99.

3 Carattere così chiamato per la forma quadra delle lettere, soprattutto a, d, e, q e per la grafia discendente e ascendente dei caratteri in senso verticale.

metà detto *bifolium*[4].

I 6 componimenti poetici e le rispettive pagine del manoscritto sono le seguenti:

- *Andreas*, incompleto (dal folio 29v al 52v del manoscritto).
- *Il Fato degli apostoli*, opera sicuramente di Cynewulf[5], poiché al folio 54 le rune inserite nel testo formano il nome Cynewulf (dal folio 52v al 54r del manoscritto).
- *Anima e Corpo I*, frammento di un testo che immagina un dialogo tra anima e corpo, lo stesso scritto lo troviamo nell' "Exeter Book" (dal folio 101v al 103v del manoscritto).
- *Frammento Omiletico I*, frammento di 47 versi allitterativi sul tema dell'inganno (dal folio 104r al 104v del manoscritto).
- *Il sogno della Croce*, corrisponde al testo in caratteri runici inciso sulla croce di Ruthwell (dal folio 104v al 106r del manoscritto).
- *Elena, lungo poema* (1321 versi) sul ritrovamento della croce di Cristo da parte di Sant'Elena, anch'esso opera autentica di Cynewulf (dal folio 121r al 133v).[6]

Tra i testi in prosa c'è una discreta varietà di temi religiosi, due omelie sul Natale (V e VI, non consecutive, divise da due poesie), vari pezzi sulle rogazioni (XI, XII e XIII in serie per i tre giorni di preghiera, X indipendente da queste tre, e XIX e XX, ancora da leggere consecutivamente ma indipendenti dalle altre prose sul tema delle Rogazioni), mentre la VIII e IX erano destinate alla prima e seconda Domenica dopo l'Epifania. La seconda metà dell'omelia XXI è di fatto una versione dell'omelia II, questo suggerisce che il copista della collezione possa aver lavorato sovrappensiero ripetendo inavvertitamente il materiale, tuttavia nel complesso il codice è ben strutturato e riesce a coprire la maggior parte delle ricorrenze religiose.[7]

Il Vercelli Book ha subito diversi danni nel corso dei secoli, mancano la

---

[4] Cfr. A.M. Luiselli Fadda, *Tradizioni manoscritte e critica del testo nel Medioevo germanico*. Laterza. Bari. 1999. p. 44.

[5] Cynewulf poeta anglosassone, autore di quattro poemi conservati in due manoscritti, Exeter Book e Vercelli Book, che ne ripostano il nome in caratteri runici. Appartenente al sec 8° o al principio del 9°. (Treccani, voce Cynewulf)

[6] Cfr. A.M. Luiselli Fadda Op. cit. p. 49.

[7] Cfr. Richards, Mary P. *Anglo Saxon Manuscripts: Basic Readings* New York: Routledge, 2001. pp. 318-321.

copertina originale e le rubriche di apertura e chiusura, inoltre alcune pagine mancano o sono state corrotte da un agente chimico che le ha rese illeggibili o quasi. L'ordine in cui è stato ritrovato il codice non è quello originale, nel corso dei secoli deve essere stato smembrato e riformato (presumibilmente per il cambio di copertina, o durante i vari cataloghi da parte dei monaci Eusebiani) alterando lo stato originale del manoscritto, che secondo alcuni studi al tempo della stesura doveva avere questa fisionomia:

Gruppo A		Omelie I-IV
Gruppo B	1a	Omelia V
	1b	Poema *Andreas* e il *Fato degli Apostoli*
	2a	Omelie VI-X
	2b	Omelie XI-XIV
	3	Omelie XV-XVIII
	4a	*Poesie Anima e Corpo*, *Frammento Omiletico I* e *Sogno della Croce*
	4b	Omelie XIX-XXI
	4c	Omelia XXII
Gruppo C		*Poema Elena* e l'omelia XXIII[8]

Lo scriba del Vercelli Book non era interessato alla normalizzazione della lingua e per la maggior parte fece una copia meccanica rispettando scrupolosamente i testi da copiare in tutte le loro specificazioni linguistiche.

Origini del Vercelli Book e teorie sul suo arrivo in Italia

Il dibattito sulle origini e sul trasferimento del Vercelli Book dall'Inghilterra all'Italia del nord è stato lungo e nel corso degli anni si sono susseguite diverse teorie che datano l'arrivo a Vercelli del manoscritto in un ampio lasso di tempo che va dal IX al XVI secolo, alcune di esse sono molto improbabili come quella dell'archeologo Costanza Gazzera che riconduce l'arrivo del codice a Vercelli a Giovanni Scoto Eriugena, filosofo irlandese, o quella del linguista Max Förster che attribuisce a Giovanni Francesco Bonomo, vescovo di Vercelli

[8] Cfr. Richards, Mary P. Op. cit. p. 326.

del XVI secolo, il merito di aver portato il codice nelle biblioteche Vercellesi. Altre teorie datano il codice più indietro nel tempo, molti pensavano che fosse arrivato insieme al Cardinal Guala Bicchieri di ritorno da uno dei tanti viaggi come diplomatico per la Chiesa durante il XIII, questa teoria e le precedenti sono state messe in dubbio da Richard Wülker supportato da S.J Herben il quale studiando il manoscritto e la sua grafia colloca il manoscritto nell'XI secolo smentendo di fatto tutte le teorie precedenti[9]. Alcuni studiosi pensano che il codice sia arrivato direttamente a Vercelli, altri invece che sia arrivato indirettamente sostando in monasteri continentali che avevano connessioni con l'Inghilterra come San Gallo[10], Bobbio[11], Fulda[12] o Wörtzburch.[13]

La maggior parte di queste teorie è invalidata dallo studio stesso del manoscritto, che come notò Wülker rivela grazie alla grafia il periodo storico in cui fu scritto, altre teorie sono invece confutate o confermate dallo studio storico della città di Vercelli nel medioevo.

Vercelli fu un importante centro fin dagli inizi della religione cristiana, subito dopo la fine delle persecuzioni cristiane fu eretta la cattedrale di Sant'Eusebio[14] in onore del primo arcivescovo della città attorno alla quale sorsero scuole e scriptoria[15] espandendo l'influenza della cattedrale sull'intera regione per più di cento anni, con alcune interruzioni dovute a diverse incursioni tra cui quella del 899 da parte delle tribù Ungheresi che provocò la distruzione della cattedrale e la perdita di molti documenti importanti. Vercelli era situata sulla strada principale che dalle Alpi portava verso Roma, posizione strategica che favorì in maniera sostanziale lo sviluppo della città e dei traffici grazie ai commercianti, pellegrini, laici e clericali che diretti verso la capitale facevano

---

9 Cfr. M. Halsall. *Vercelli and the Vercelli Book, Modern language association*. *PMLA.* 84. 1969. p. 1545.

10 San Gallo è una cittadina Svizzera fondata nel 7 secolo, qui aveva sede un importante abbazia, oggi patrimonio dell'UNESCO

11 Bobbio piccolo comune in provincia di Piacenza, importante durante il medioevo grazie all'abbazia di San Colombano.

12 Fulda è una cittadina tedesca situata lungo il fiume Fulda

13 Wörtzburch città nella regione della Franconia.

14 Tra il V e il VI secolo fu edificata una basilica paleocristiana che giunse integra fino alla fine del 1570 quando se ne iniziò la demolizione; nel suo massimo splendore la cattedrale aveva cinque navate divise da colonne con transetto, ed era preceduta da un ampio portico; l'abside era decorata a mosaico, e nel presbiterio era collocato l'ambone scolpito da Benedetto Antelami. (Wikipedia, voce duomo di Vercelli).

15 Scriptoria il locale dei monasteri nel quale era organizzata ed eseguita l'opera di trascrizione dei codici.

tappa a Vercelli per brevi o lunghi periodi, arricchendo cosi le tasche dei locali. Col passare degli anni Vercelli divenne un grande centro religioso e l'intera città era sotto la supervisione dell'ordine degli Eusebiani, tranne l'abbazia di Sant'Andrea che si rese indipendente tramite un decreto papale, sotto il diretto controllo degli Eusebiani fino alla fine del XVI secolo c'era anche l'ospizio di Santa Brigida degli Scoti meta di molti viaggiatori che sostavano qui durante i loro pellegrinaggi.

Nel 1343 l'ospizio fu ceduto in via definitiva all'abbazia di Sant'Andrea come pagamento per i fondi concessi agli Eusebiani durante gli anni delle guerre tra Guelfi e Ghibellini che lasciarono gli Eusebiani con poche risorse ma prima di questa data Santa Brigida era meta continua di pellegrini molti dei quali provenienti dall'Inghilterra, l'afflusso di persone si intensificò ulteriormente nel 1228 quando grazie all'accordo con quattordici professori dall'università di Pavia si trasferirono a Vercelli fondando l'università Vercellese, i professori si stabilirono a Santa Brigida che si trasformò poi in un vero e proprio ostello messo a disposizione degli studenti di tutta Europa.

L'università prosperò fino al XII secolo poi complice il declino intellettuale e religioso di Vercelli l'università venne trasferita a Torino nel 1335 portando Vercelli alla fine del XIV secolo a essere solamente un luogo di sosta lungo la strada per Roma.

Da questa piccola analisi storica della città possiamo facilmente intuire che Vercelli è stata per vari secoli una meta importante del nord Italia, visitata da tantissimi Britannici, non è quindi sorprendente il ritrovamento del Vercelli Book all'interno delle biblioteche Eusebiane, ma vista la mancanza di firme sul documento o altre prove certe è impossibile attribuirlo a una specifica persona come hanno cercato di fare Gazzera, Förster, o gli studiosi che volevano il codice portato da Guala Bicchieri[16].

Non abbiamo la possibilità di identificare né lo scriba del Vercelli Book né il suo proprietario a Vercelli sappiamo però che il manoscritto è stato redatto prima dell'XI secolo e questa certezza ce la forniscono diversi cataloghi ritrovati all'interno delle biblioteche Eusebiane.

Il catalogo che riporta al suo interno tutti gli inventari fatti nei secoli mostra

---

16 Cfr. M. Halsall, Op. cit. p. 1545.

diverse annotazioni che fanno presupporre la presenza del Vercelli Book all'interno degli archivi; uno dei più chiari è quello redatto dal canonico Leone che alla riga novanta scrive : *Liber Gothicus sive Longobardus (eum legere non valeo)* , (Halsall 1969, p. 1547) "Libro in Gotico o Longobardo (non riesco a leggere)"; altro riferimento al Vercelli è presente nell'inventario del 1750 redatto da Giuseppe Bianchino che riporta: *CXVII sæculi X. Liber ignotæ linguæ. Videtur liber Homiliarius per anni cinculum, Ut constat ex nonnulliss rubricis latine conscriptis (linguæ theotiscæ).* (Halsall 1969, p. 1547) Qui Bianchino descrive un libro in lingua sconosciuta del X secolo.
Questi cataloghi possono far presumere la presenza del Vercelli Book all'interno degli archivi Eusebiani e della sua redazione nel X secolo, la certezza di queste ipotesi è data da due cataloghi importantissimi per collocare il codice a Vercelli e nel tempo giusto.
Il primo catalogo è datato 1426 ed è formato da 89 fogli di carta raccolti all'interno di 18 quaderni, il catalogo è redatto da un canonista e un notaio locale che marchiò i quaderni con cinque marchi (grazie ai quali possiamo risalire alla data del catalogo). È un elenco di 90 volumi conservati all'interno della chiesa, i quali vengono accuratamente descritti riportando descrizioni sia fisiche che del contenuto, il nome dello scriba che li ha creati e la data di quando è stata redatta la copia[17].
Durante l'analisi del catalogo alcuni studiosi del Vercelli Book hanno trovato un volume la cui descrizione è molto interessante dettagliata come le altre dal punto di vista fisico la descrizione del contenuto risulta essere molto vaga: *Item liber omeliarius antiquissimus non abens principium nec finem, et aliquantulum dequantenarus, cum asseribus aliquantunum a libro remotis, scriptus in carta.* (Halsall 1969, p. 1549). Il termine *antiquissimus* non si riferisce all'età del manoscritto, ma più probabilmente all'incapacità da parte del cataloghista di decifrare il testo, il che porta il cataloghista a identificarlo come uno dei più antichi, inoltre viene descritto come un libro scritto in pergamena senza copertina[18], rubriche di apertura e di chiusura[19], e con i quaderni smembrati[20]

---

17 Cfr. M. Halsall, Op. cit. p. 1548.

18 Probabilmente sono andati persi i lacci, e sono stati rimpiazzati nel tempo perché nel 1768 il VB viene descritto minuziosamente e non si accenna alla copertina mancante.

19 Il cataloghista del XV secolo che ha redatto il catalogo scriveva sempre l'incipit e la fine dei manoscritti catalogati, quindi al tempo del catalogo la prima pagina era già stata cancellata si

caratteristiche che come abbiamo visto in precedenza appartengono al codice.

Un altro indizio a supporto che il libro catalogato sia il Vercelli Book è una nota aggiunta da un lettore del XVIII secolo *Omiliarum liber ignotis idiomatis* ovvero libro di omelie in idioma sconosciuto.

L'altro catalogo che fa luce sul manoscritto è del XVIII secolo e ne esistono due versioni: una scoperta da poco all'interno degli archivi Eusebiani l'altra in possesso del Dott. Ernesto Gorini di Vercelli.[21]

La prima versione consiste in un singolo quaderno formato da nove fogli di carta non rilegati, piegati e numerati da 1 a 31 dal titolo *Recensio Codicum Msrum qui in Tabulario Vercellensis Ecclesiæ Asservantur*; (Halsall 1969, p. 1548) la seconda versione è invece intitolata *Recensio Codicum Centum MSS. ex is, qui in Tabulario Vercellensis Ecclesiæ Asservantur* (Halsall 1969, p. 1548) formata da 20 fogli raccolti in 3 quaderni da 6 fogli ciascuno e un quaderno formato da 2 fogli.

La versione originale è quella scoperta di recente e nelle pagine che vanno dalla 18 alla 25 il cataloghista si è impegnato in una dettagliata e minuziosa descrizione personale del manoscritto, dei suoi contenuti e di come potevano essere decifrati, questa descrizione fornisce una datazione certa del Vercelli Book invalidando di fatto molte delle tesi viste in precedenza[22].

Grazie a tutti questi indizi abbiamo la certezza che il codice sia stato redatto nel tardo X secolo, mentre per capire il periodo in cui sia arrivato a Vercelli ci viene in aiuto lo studio del testo, il quale contiene numerose annotazioni in lingua inglese; a pagina 24v troviamo però in un verso del salmo XXVI una nota scritta in grafia italiana del XII secolo, possiamo quindi supporre che sia questo il periodo in cui il manoscritto fu portato in Italia da uno dei tanti viaggiatori, per poi essere ritrovato all'interno degli archivi cattedrali da uno studente che non riusciva a catalogarlo nel XIX secolo[23].

---

capisce che mancano queste pagine in quanto la prima pagina non avrebbe potuto vantare più di un H di apertura e "Amen fiat" è ambiguo a pagina 135v.

[20] Mancano le pagine 42v, 55v, 63v, 75v, 83v, 85v, 97v, 100v, 103v, 111v.

[21] Cfr. M. Halsall, Op. cit. p. 1548.

[22] Cfr. M. Halsall, Op. cit. pp. 1548-1549.

[23] Cfr. M. Halsall, Op. cit. pp. 1549-1550.

Capitolo 2

Digitalizzare un documento

Vantaggi del formato digitale e le caratteristiche da rispettare

Il Vercelli Book si è rivelato un manoscritto di grande importanza nello studio dell'inglese antico, la richiesta a poter visionare il codice è cresciuta, diventando il centro di molti studi; questo grande interesse da parte degli studiosi ha reso necessaria la diffusione del documento in maniera più fruibile a chi voglia studiare il manoscritto e abbia bisogno di poter fare una traduzione linea per linea, fondamentale per capire la grammatica e la metrica dei poemi.

Per rendere il Vercelli Book più accessibile è necessario che venga digitalizzato. Un testo è definibile in formato digitale quando ogni parte del testo (introduzione, edizione del testo, critica, note etc....) è disponibile in digitale, possibilmente organizzato come un ipertesto[24] e che contenga al suo interno immagini del manoscritto originale oggetto della digitalizzazione (indispensabile nel caso del Vercelli Book). Il formato digitale grazie alle sue caratteristiche permette una diffusione e consultazione del codice, che col formato cartaceo non si possono ottenere:

Conservazione: i bit non occupano uno spazio fisico se non quello del supporto che li ospita; le edizioni digitali quindi non danno problemi di ingombro, ma se il problema della conservazione nello spazio è irrilevante è invece un problema la conservazione nel tempo; a causa della rapida obsolescenza tecnologica testi di pochi anni rischiano di diventare inaccessibili in breve tempo, per questo è importante adottare formati standard che diano resistenza nel tempo, formati come ASCII e XML garantiscono una buona durata dell'informazione.

Trasmissibilità: il libro tradizionale è soggetto a tutti i problemi della distribuzione fisica, un'edizione digitale invece ha nella grande trasportabilità uno

[24] Un ipertesto è un testo organizzato in un insieme di moduli elementari che ne rende possibile la lettura, integrale o parziale, secondo diversi percorsi logici, scelti dal lettore in base a sue personali esigenze. (Treccani, voce ipertesto).

dei suoi più grandi vantaggi, può essere trasmessa da un capo all'altro del pianeta in pochi secondi e a costi sensibilmente inferiori rispetto a una versione cartacea.

Riproducibilità: il digitale fornisce importanti possibilità alla diffusione, la copia digitale è sempre disponibile e dove non vi sono problemi di diritti d'autore è facilmente raggiungibile da chiunque voglia usufruire del contenuto.[25]

Il successo di un edizione digitale non dipende solamente dalle sue caratteristiche fisiche, ha infatti un impatto fondamentale il modo in cui viene strutturata e organizzata all'interno di un software, il quale deve rispettare dei parametri oppure rischia di essere inutilizzabile. I parametri fondamentali che un software per la visualizzazione di edizioni digitali deve avere come obbiettivo sono:

Accessibilità: tutti coloro che vogliono interagire sul testo in formato digitale devono essere in grado in maniera semplice. L'accessibilità è importante per dare le stesse opportunità a tutti gli utilizzatori dell'edizione.

Espandibilità: il documento una volta digitalizzato non è indipendente, deve essere in grado di estendersi integrando nuove funzionalità come la multimedialità e l'interazione con l'utente. L'edizione digitale può essere per esempio integrata con suoni ed immagini. Essendo un oggetto informatico, il testo digitale può essere trattato con strumenti della tecnologia informatica, sono quindi innumerevoli gli usi cui può essere adattato dai software per edizioni critiche, dai motori di ricerca alle applicazioni per la lettura dei testi paralleli.

Usabilità: per non rendere l'esperienza dell'utente frustrante l'edizione digitale deve essere raggiungibile in modo trasparente e semplice. L'obbiettivo di una buona usabilità è quello di rendere la copia digitale una valida alternativa alla copia stampata; questo compito è molto difficile e con la tecnologia attuale non è possibile adempiere il compito totalmente. Per svolgere il compito nella miglior maniera possibile il software deve rispettare alcune condizioni in fase di progettazione:

- **Velocità delle operazioni**: il sistema deve risponde ai comandi dell'utente in maniera rapida.

---

[25] Cfr. Vantaggi e limiti del libro elettronico http://www.digisic.it/documentazione/codifica_baltico/html/II.7.html (Visitato il 12 Marzo 2015).

- **Robustezza**: il sistema deve essere il più tollerante possibile agli errori dell'utente.
- **Capacità di ripresa**: il sistema deve essere in grado di riprendersi dagli errori dell'utente.
- **Adattabilità**: il sistema dovrebbe essere in grado di distaccarsi da un singolo modello di lavoro.
- **Velocità di apprendimento**: l'utente deve essere in grado di padroneggiare il sistema in breve tempo.[26]

Se un edizione digitale rispetta questi parametri sarà performante e fornirà all'utente uno strumento valido per poter studiare il codice.

Software per esplorare l'edizione digitale

Chiariti i vantaggi di un'edizione digitale e soprattutto le caratteristiche a cui un'edizione digitale deve ambire, per poter essere un valido strumento di studio, possiamo iniziare a pensare a quale tipo di software sviluppare per caricare il testo digitalizzato.

Per la realizzazione di un edizione digitale non esiste un modello standard, un programmatore ha a sua disposizione moltissimi strumenti per poter realizzare la struttura del sistema, ma la prima scelta che deve sempre fare è se realizzare un software *ad hoc* oppure se affidarsi agli strumenti messi a disposizione dai web browser.

Ad oggi la scelta più naturale è quella di sviluppare su web browser (o su programmi compatibili con capacità ipertestuali); il World Wide Web è il migliore e più diffuso mezzo per condividere documenti di tutti i tipi, ma la distribuzione via web non è sempre possibile, possono esserci problemi tecnici, o problemi burocratici, come la mancata concessione dei diritti da parte dei possessori a pubblicare in rete immagini o testi, infatti molte edizioni digitali stand-alone sono attualmente basate sui browser e potrebbero essere pubblicate in rete, ma non vi possono essere inserite per la mancanza di autorizzazioni. Una edizione digitale basata sui browser presenta numerosi vantaggi, non richiede nessun tipo di installazione, piccoli rischi di incompatibilità con i sistemi operativi, e l'interfaccia

[26] Cfr. Sommerville Ian. 2004. *Software Engineering* p. 48. http://ifs.host.cs.st-andrews.ac.uk/Books/SE7/Presentations/PDF/ch16.pdf (Visitato il 28 Marzo 2015).

è familiare all'utente; tuttavia ci sono delle problematiche, sono fortemente condizionati dal linguaggio di programmazione con cui vengono creati, infatti possono essere inflessibili e di difficile personalizzazione; un problema limitante soprattutto per le edizioni digitali basate su immagini, che necessitano di molte funzionalità, alcune delle quali devono essere implementate tramite plug-in esterni non sempre facilmente integrabili nel sistema. Il secondo problema è che lo sviluppatore generalmente ha in mente un browser specifico, e questo può portare all'inutilizzabilità da parte dell'utente su differenti sistemi operativi o versioni del browser troppo vecchie.

Le edizioni digitali disegnate per software appositi invece offrono una grande flessibilità donando al programmatore una maggiore libertà per implementare differenti funzionalità e impostare l'interfaccia grafica esattamente come desidera, ma questa flessibilità viene controbilanciata da un maggiore rischio di incompatibilità, e di portabilità attraverso diverse piattaforme sia software che hardware e di fatto un software dopo cinque anni diventa obsoleto e incompatibile con la maggior parte delle macchine.

Questi problemi sono troppo limitanti rispetto ai vantaggi che porta un software specifico, per tali motivi è largamente consigliato l'utilizzo di una programmazione su browser web.[27]

Criteri e caratteristiche da rispettare in una interfaccia grafica per le edizioni digitali

Deciso il tipo di software che vogliamo realizzare per la navigazione dell'edizione digitale, inizia una fase detta di *brainstorming* in cui il programmatore pensa a come dovrà essere il prodotto finito, quali funzionalità dovrà implementare e come dovrà apparire l'interfaccia grafica agli occhi dell'utente.
Questa fase è molto importante perché influenzerà tutto il lavoro di programmazione vera e propria; durante la progettazione dell'interfaccia grafica c'è da tenere conto di molti fattori che devono essere rispettati per avere una edizione digitale fruibile dall'utente in maniera adeguata.

---

[27] Cfr. Rosselli del Turco Roberto. *After the editing is done: Designing a Graphic user interface for digital edition.* Digital Medievalist. 2011. http://www.digitalmedievalist.org/journal/7/rosselliDelTurco/ (Visitato il 07 Aprile 2015).

I punti fondamentali che un programmatore deve rispettare durante l'elaborazione di un interfaccia grafica sono:[28]

Facilità di utilizzo: l'interfaccia dovrebbe usare termini e concetti familiari all'utente medio che utilizzerà il sistema.

Coerenza: l'interfaccia dovrebbe essere il più coerente possibile, operazioni simili dovrebbero attivarsi nello stesso modo; azioni differenti dovrebbero attivarsi in maniera differente. Il menu di navigazione deve avere un indice chiaro e conciso; strumenti come menu a tendina o strumenti per la selezione di testi e immagini aiutano a rendere la navigazione intuitiva, ma dobbiamo stare attenti a non rubare troppo spazio ai contenuti, soprattutto per una edizione digitale basata su immagini o testi (in particolar modo se si confrontano due o più testi) lo spazio riveste un fattore fondamentale e deve essere usato con saggezza, il modo più semplice per non rubare troppo spazio ai contenuti è creare un applicazione che ci permetta di visualizzare i contenuti a tutto schermo, nascondendo ogni altro componente.

Linearità: l'utente non dovrebbe mai sorprendersi del comportamento del sistema. Se un comando è conosciuto in una data maniera, l'utente dovrebbe essere in grado di capire l'operazione di un comando simile, anche non avendolo mai utilizzato.

Ripristino: l'interfaccia dovrebbe includere meccanismi per permettere all'utente di rimediare ai propri errori. Possono essere inclusi comandi per tornare indietro, tasti di conferma per operazioni distruttive, etc.

Guida per l'utente: quando si verifica un errore il sistema dovrebbe provvedere a un feedback per aiutare l'utente a non ripeterlo. I messaggi di errore vanno studiati accuratamente in fase di progettazione perché sono di fondamentale importanza; messaggi poveri o troppo lunghi possono significare che un utente si rifiuti di ascoltare il sistema. Per avere effetto i messaggi devono essere educati, concisi, coerenti e costruttivi, esistono delle linee guide che aiutano un programmatore a realizzare messaggi in maniera adeguata:

---

[28] Cfr. Rosselli del Turco Roberto. 2011. Op. cit. http://www.digitalmedievalist.org/journal/7/rosselliDelTurco/ (Visitato il 12 Marzo 2015).

- **Contesto:** il sistema dovrebbe essere a conoscenza di ciò che l'utente stia facendo e dovrebbe generare messaggi rilevanti per l'attività corrente.
- **Esperienza:** per un utente esperto lunghi messaggi possono essere irritanti, tuttavia i principianti trovano di difficile comprensione le dichiarazioni troppo brevi per questo è necessario dare all'utente l'opportunità di poter scegliere quale tipo di messaggio ricevere.
- **Stile:** i messaggi dovrebbero essere positivi, non devono contenere insulti o cercare di essere divertenti.
- **Cultura:** il progettista dove possibile deve avere familiarità con la cultura del paese in cui verrà utilizzato il sistema. Ci sono differenze culturali forti tra Europa, America, Asia. Un messaggio adatto per una cultura potrebbe non esserlo per un'altra.[29]

Diversità: l'interfaccia dovrebbe fornire adeguate strutture di interazione per diversi tipi di utente. Un utente con problemi alla vista dovrebbe avere la possibilità di ingrandire il testo, etc. Questi punti sono i principi base che ogni sviluppatore di interfacce grafiche deve seguire per realizzare un prodotto di qualità, tuttavia per un'edizione digitale basata sulle immagini i principi base non bastano e si deve tenere conto di altri fattori dovuti alle particolari funzionalità richieste da questo tipo di edizioni digitali.

Buone funzionalità ipertestuali: un'edizione digitale basata sulle immagini fa un massiccio uso di questo tipo di funzionalità, le più comuni includono la ricerca di una parola o di un oggetto, la lente di ingrandimento, diverse modalità di lettura, note editoriali, link a specifiche regioni o a immagini associate. Implementare tutte queste funzioni rappresenta la vera sfida per i programmatori che desiderano raggiungere un'elevata qualità in coerenza, facilità di utilizzo, e linearità.

Gestione caratteri speciali: le edizioni su testi storici o in lingue diverse dall'inglese, possono far uso di caratteri speciali non rintracciabili nei comuni font dei sistemi operativi o nello standard Unicode; in questi casi bisogna prestare attenzione al modo in cui l'utente può avere accesso a questo tipo di caratteri.

---

[29] Cfr. Sommerville Ian. Op. cit. p. 30.
http://ifs.host.cs.st-andrews.ac.uk/Books/SE7/Presentations/PDF/ch16.pdf
(Visitato il 28 Marzo 2015).

Strumenti per la manipolazione delle immagini: spesso c'è il bisogno di manipolare le immagini, questo può avvenire in modo elementare come la possibilità di zoomare precise aree o ingrandire l'immagine a tutto schermo, oppure possono essere necessarie funzioni più complesse; come il poter applicare dei filtri alle immagini (cambio colore, contrasto, luminosità) ai fini di ricerca specifici; implementare funzioni così complesse può andare ad intaccare tutti i principi base visti precedentemente, per evitare questo è uso comune inserire solo gli strumenti base come parte della grafica, ma dare la possibilità di un accesso diretto alle immagini in alta qualità per poterle manipolare in maniera stand-alone.

Avanzate funzioni di ricerca: nel caso in cui il testo sia codificato in eXtensible Markup Language(XML)[30], possiamo avere la possibilità di effettuare complesse operazioni di ricerca sui singoli termini e brani di testo più lunghi oggetto di marcatura.

Strumenti supplementari (glossario, concordanze etc.): nelle edizioni digitali basate su immagini o testi ci sono una grande quantità di informazioni e i progettisti devono considerare il modo in cui l'utente può accedere a tutte queste informazioni, mettendogli a disposizione una vasta gamma di strumenti complementari, tra cui glossari concordanze, appendici, archivi dei materiali. Un programmatore si deve domandare se ha senso replicare i modelli di stampa tradizionale e integrarli nell'ambiente digitale; le decisioni prese su questo punto hanno un impatto notevole sui requisiti generali del sistema, e per avere un buon risultato deve rispettare alcune regole:

- Mai mettere troppi oggetti in primo piano.
- Consentire un facile passaggio da un oggetto all'altro.
- Consentire un facile sistema per massimizzare/minimizzare, nascondere/mostrare.
- Consentire un facile modo per selezionare un oggetto (preferibilmente tramite menu a discesa), questo principio è noto come *"principio della disponibilità costante"*.

[30] XML (eXtensible Markup Language) metalinguaggio flessibile ed efficace per la condivisione di dati tra sistemi diversi. http://www.w3.org/XML/

- Nascondere opzioni non disponibili in una data modalità.[31]

Una interfaccia grafica ben ordinata e pulita renderà la navigazione dell'edizione un'attività molto più facile.

Se un'edizione digitale viene creata cercando di rispettare tutte queste linee guida sarà ben sviluppata e in grado di soddisfare l'utente, ma prima di poter essere distribuita c'è bisogno di un periodo di *beta testing* dove il prodotto finito viene dato a un numero limitato di utenti che hanno il compito di testare l'edizione e riferire al programmatore eventuali problemi da risolvere.

Finito il periodo di beta e corretti i vari errori l'edizione digitale è pronta per essere distribuita a chiunque sia interessato.[32]

---

[31] Cfr. Sommerville Ian. Op. cit. pp. 29-32. http://ifs.host.cs.st-andrews.ac.uk/Books/SE7/Presentations/PDF/ch16.pdf (Visitato il 28 Marzo 2015).

[32] Cfr. Rosselli del Turco Roberto. 2011. Op.cit. http://www.digitalmedievalist.org/journal/7/rosselliDelTurco/ (Visitato il 11 Aprile 2015).

Capitolo 3

Il progetto EVT (Edition Visualization Technology)

Agli albori il progetto denominato EVT (Edition Visualization Technology) era una ricerca sperimentale ideata dal Professor Roberto Rosselli del Turco per gli studenti del corso di Informatica Umanistica; il progetto puntava a indagare su alcuni aspetti dell'interfaccia utente, in particolare la ricerca si soffermava sui problemi di usabilità, spesso sottovalutati in progetti simili, e tendeva ad incoraggiare l'uso dei formati standard per garantire la longevità massima all'edizione.

Data la sua natura didattica il codice sorgente è stato open source fin dall'inizio, per darne libero accesso a tutta la comunità accademica.

EVT nasce all'interno del progetto del Vercelli Book digitale che a sua volta è ispirato al progetto di Kevin S. Kiern *Electronic Beowulf* [33], un edizione per CD/DVD mirata a rendere digitale il manoscritto del X secolo *Beowulf*.[34]

I primi prototipi di EVT non riuscivano a raggiungere gli obbiettivi proposti per via di alcuni problemi: il visualizzatore aveva un interfaccia che risultava troppo ingombrante, inoltre c'erano problemi di instabilità dovuti all'implementazione di troppi *widget*[35] di origine diversa ma utilizzati contemporaneamente; altra pecca della prima versione era il caricamento dei dati direttamente all'interno del codice web, senza nessuna possibilità di configurazione.

EVT cosi come era partito non era in grado di migliorarsi e di raggiungere gli standard per cui il progetto era stato avviato, per questo il professor Del Turco insieme ai suoi collaboratori decisero di ripartire completamente da zero, focalizzandosi solamente sulle funzionalità principali, rimuovendo quelle

---

33 *Electronic Beowulf*. http://ebeowulf.uky.edu/.

34 *Beowulf* poema epico, il più antico tramandatoci delle letterature germaniche. Composto intorno al 700 da un Anglo, pervenutoci, trascritto da amanuensi del Wessex, in un unico manoscritto di fine 10° o primi dell'11° sec. Consta di 3183 versi. Narra le gesta dell'eroe Beowulf che uccide il mostro Grendel e successivamente un drago, perdendo la vita nella seconda impresa (Treccani, voce Beowulf)

35. In programmazione un widget è un componente grafico dell'interfaccia utente di un software.

secondarie[36]; l'ispirazione per il cambiamento è stata presa da progetti simili[37] sviluppati all'interno della comunità TEI[38].

Il problema del caricamento dei dati all'interno del codice è stato risolto costruendo l'edizione digitale intorno ai dati stessi, il file TEI XML è stato il punto di partenza intorno al quale è stato poi costruito il tutto; in questo modo l'editor può concentrarsi unicamente sul proprio lavoro: la trascrizione del testo, e lo fa utilizzando pochi strumenti di configurazione.

Grazie all'utilizzo di file XML c'è stata la possibilità di fare test su altre edizioni già pronte, per verificare se EVT potesse andare oltre lo specifico strumento per la visualizzazione del Vercelli Book e diventare uno strumento universale per la visualizzazione di edizioni codificate in TEI XML, comprensive di immagini.

Con questo nuovo approccio sono stati raggiunti due risultati molto importanti, prima di tutto EVT è semplice da utilizzare, l'utente applica un foglio di stile XSLT[39] e, una volta caricato questo appare come una edizione *web-ready*; in secondo luogo l'edizione web che è il prodotto del caricamento del file è basata su un'architettura solo client che non richiede nessun tipo di server, avere un'architettura lato client vuol dire che può essere copiato ovunque (un server web, cloud, mezzi fisici), e garantisce un funzionamento corretto e longevo su tutti i più recenti web browser (Internet Explorer, Mozilla Firefox, Chrome, Safari e Opera).

L'architettura è stata realizzata utilizzando tecnologie web aperte e standard come HTML[40], CSS[41], Javascript[42] e jQuery[43]; mentre per ridurre al minimo futuri

---

36 Cfr. Roberto Rosselli del Turco. *Edition Visualization Technology: A simple tool to visualize TEI-based digital edition.* Journal of the text encoding initiative. 2013. *http://jtei.revues.org/1077* (Visitato il 12 Marzo 2015).

37 Il *TEI Boilerplate*, collezione John A. Walsh di fogli di stile XSLT http://dcl.ils.indiana.edu/teibp/, e il lavoro di Solenne Coutagne per la "*Berliner Intellektuelle 1800-1830* "project.10. http://jtei.revues.org/pdf/707

38 Text Encoding Initiative (TEI) è un consorzio che sviluppa standard per la rappresentazione di testi in formato digitale. http://www.tei-c.org/index.xml

39 XSLT è un linguaggio per trasformare i documenti XML in altri documenti XML, progettato per essere utilizzato come parte di XSL. XSL specifica lo stile di un documento XML utilizzando XSLT, descrivendo come quest'ultimo si trasforma utilizzando un vocabolario di formattazione.

XSLT può anche essere usato in modo indipendente da XSL ma è stato progettato principalmente per i tipi di trasformazione che sono necessarie quando XSLT viene utilizzato come parte di XSL. (W3C, voce xslt). http://www.w3.org/TR/xslt

40 HTML è il linguaggio di marcatura utilizzato per la creazione di pagine web. http://www.w3.org/html/

41 CSS indica un insieme di regole per determinare l'aspetto grafico di un documento generato con un linguaggio a marcatori come HTML. http://www.w3.org/Style/CSS/

problemi di compatibilità è stata creata in modo modulare, cosi che ogni componente possa essere sostituito indipendentemente dagli altri moduli.
Al fine di ottenere un prodotto longevo i plug-in sono stati scelti tra gli open source con maggior supporto sperando in un aggiornamento costante e duraturo da parte della comunità.
La struttura modulare ha permesso di creare uno strumento molto facile che richiede pochissimo lavoro e nessuna conoscenza da parte dell'utente, a parte il saper utilizzare l'editor XML.
EVT è basato su un unico foglio di stile (evt_builder.xlt) il quale avvia una serie trasformazioni XSLT chiamando a turno tutti gli altri moduli, che appartengono a due categorie: quelli dedicati alla costruzione del sito HTML, e quelli di trasformazione XML (inseriti tutti all'interno della cartella builder_pack) i quali estraggono il testo dell'edizione.
Seguendo tre semplici passi possiamo creare qualsiasi edizione digitale l'unico requisito è che i fogli di stile soddisfino i criteri del progetto:

- Copiare i dati nella cartella/input_data (al cui interno troviamo diverse sotto cartelle per il testo e le immagini).
- Opzionalmente si possono modificare le impostazioni di trasformazione modificando **evt_builder-conf.xsl**, per specificare ad esempio la presenza di immagini.
- Applicare il foglio di stile **evt_builder.xsl** al documento XSL TEI usando un editor XML come Oxygen o un altro editor simile (Eclipse XSD editor, Xmlfox, EditiX, Altova Xml Spy); che abbia un processore per XSLT 2.0.

Quando la trasformazione XSLT è finita il punto di partenza per l'edizione digitale è il file **index.html** nella directory principale; tutte le pagine HTML risultanti dalle trasformazioni vengono memorizzate nella cartella output_data, e visto che tutti i file vengono generati dinamicamente è possibile modificare le configurazioni o cancellare tutto per poi ricominciare, ogni volta i file verranno ricreati nei posti predefiniti.

---

[42] Javascript è un linguaggio di programmazione orientato agli oggetti, utilizzato per arricchire di funzionalità i siti web. http://www.w3.org/standards/webdesign/script.html

[43] È una libreria di funzioni Javascript per le applicazioni web, ha l'obiettivo di semplificare la manipolazione e la gestione delle pagine HTML. https://jquery.com/

Allo stato attuale EVT può creare edizioni digitali basate su immagini con due possibili livelli di edizione: diplomatico e interpretativo; ciò significa che una trascrizione codificata utilizzando elementi appartenenti al modulo 13 TEI[44] è compatibile, o al massimo richiede piccole modifiche, per poter essere visualizzato su EVT.[45]

Lo schema di trascrizione del Vercelli Book si basa sullo schema TEI Standard senza personalizzazioni o attributi aggiuntivi.

Una volta creata la pagina **index.html** e lanciata su qualsiasi browser, l'utente si troverà davanti la visualizzazione predefinita, che consiste nell'immagine del manoscritto sul lato sinistro e il testo corrispondente sul lato destro, in alto a destra sono disponibili le icone per poter cambiare la visualizzazione del manoscritto, le viste disponibili sono:

Figura 1. Viste disponibili EVT.

- **Immagine-Testo:** è la schermata di default che mostra l'immagine di una pagina del manoscritto e il testo corrispondente a uno o più livelli di edizione.[46]
- **Testo-Testo:** questa schermata è concepita per confrontare diversi livelli di edizione, che possono essere scelti attraverso il menu a tendina nella barra degli strumenti.
- **Modalità libro:** questa vista espande il frame dell'immagine per mostrare su entrambi i lati il manoscritto, in questa modalità vedremo il verso di un folio e il folio successivo dal lato retto. Per adesso non è possibile visualizzare i fogli singoli e visualizzarli in maniera ordinata, questo perché l'editor codifica i numeri dei fogli mediate il tag `<pb>` che include le lettere "v" ed "r" per marcare retto e verso delle pagine, in questo modo EVT assocerà automaticamente ogni foglio, per le immagini all'interno

---

44 Il modulo 13 TEI è consultabile all'indirizzo: http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ND.html.

45 Cfr. Roberto Rosselli del Turco. 2013. Op. cit. *http://jtei.revues.org/1077* (Visitato il 12 Marzo 2015).

46 La funzione è ispirata al software di Martin Holmes: Markup Tool, ed è implementato all'interno del codice XSLT e CSS. http://sourceforge.net/projects/imagemarkuptool/

della cartella imput_data/ images, seguendo uno schema di denominazione "verso-retto" (ad esempio: 104v-105r. png).

Anche se le diverse modalità di schermata accedono a diversi tipi di contenuti gli algoritmi di navigazione utilizzati permettono all'utente di passare facilmente da una modalità all'altra senza perdere la posizione di navigazione corrente.
Tutti i contenuti sono mostrati all'interno del frame HTML, non importa quale schermata sia caricata, possiamo ingrandire il fotogramma per mettere il focus su un contenuto specifico, nascondendo temporaneamente le altre componenti dell'interfaccia utente, è anche possibile rimpicciolire le barre degli strumenti per aumentare lo spazio dedicato alla visualizzazione dei contenuti.
EVT dispone di diversi strumenti per manipolare ed esaminare le immagini dei manoscritti, oltre alla funzione di zoom sempre attiva nella barra degli strumenti si trovano altre funzioni:

Figura 2. Strumenti di manipolazione EVT.

- **Lente di Ingrandimento:** una lente di ingrandimento per esplorare nel dettaglio i fogli del manoscritto, che mostra un area di una versione ad alta risoluzione della stessa immagine, ciò comporta un maggior dettaglio rispetto allo zoom standard.
- **HotSpot:** attivandola sulle immagini, se presenti, vengono visualizzate le note o i dettagli, le quali se selezionate aprono dei popup con la descrizione. Questa caratteristica non è pienamente funzionante ed è possibile che venga sostituita in futuro.
- **TextLink:** attivandola vengono evidenziati i collegamenti tra l'immagine del manoscritto e le righe del testo corrispondente.
- **Miniature:** attivandola avremo un'anteprima in miniatura di tutte le immagini digitalizzate del manoscritto.

EVT con queste caratteristiche è entrato in fase di beta il 24 Dicembre 2013 e grazie all'aiuto della comunità in poco tempo sono stati risolti piccoli bug[47] che

[47] Errore nella scrittura di un programma software.

hanno permesso il rilascio ufficiale su Sourceforge[48] un portale di diffusione per progetti open source.[49]

Futuri Sviluppi

Nonostante il rilascio ufficiale lo sviluppo di EVT non si è fermato e in progetto ci sono molti cambiamenti, alcuni dei quali a lungo termine che potrebbero richiedere il totale cambio dell'architettura utilizzata fino a questo momento.

Le nuove modalità e funzioni riguardano la grafica, il motore di ricerca e l'implementazione di Digital Lightbox.
La grafica attuale di EVT è soddisfacente per le funzioni implementate finora, ma con l'inserimento di nuove caratteristiche la grafica dovrà essere modificata per poter accogliere in maniera adeguata le nuove funzioni, lavorare sulla grafica di EVT non è complicato data la sua solida base e la notevole elasticità, questo permette di cambiare e adeguare la grafica in modo semplice.
Uno degli aggiornamenti più importanti in sviluppo è *EVT Search* un motore di ricerca libera e per parole chiave; in fase di progettazione per poter integrare una funzione di questo tipo è stato proposto di passare a un architettura lato server prendendo spunto da database XML open source, ma per il momento è stato deciso di mantenere EVT con un'architettura lato client.
Abbandonata l'idea di un server sono stati visionati i plug-in più popolari programmati in Javascript, questa ricerca ha portato a due risultati:

- **Tipue Search**: è un motore di ricerca sviluppato e rilasciato dal MIT[50] finalizzato alla ricerca e all'indicizzazione di grandi raccolte.
  Utilizza Javascript e JSON[51] per memorizzare il contenuto; il programma accede alla struttura dati, fa una ricerca sui file e ne restituisce i risultati.
  Tipue Search può funzionare in diverse modalità, per gli scopi di EVT è sufficiente la modalità statica che non necessita di alcun server ma fa ricerca sui contenuti memorizzati all'interno del file tipuedrop_content.js.

---

48 Il portale Sourceforge è disponibile all'indirizzo: http://sourceforge.net/projects/evt-project/.

49 Cfr. Roberto Rosselli del Turco. 2013. Op. cit. *http://jtei.revues.org/1077* (Visitato il 12 Marzo 2015).

50 Massachusetts Institute of Technology. http://web.mit.edu/

51 Javascript Object Notation si basa su un sottoinsieme del linguaggio di programmazione Javascript. http://www.w3.org/TR/json-ld/

Per poter far funzionare il plug-in in modo corretto è stato necessario modificare alcuni fogli di stile per la trasformazione JSON, l'output delle trasformazioni consiste in due file diplomatic.json che contiene il testo dell'edizione diplomatica del Vercelli Book, ed il file facsimile.json che contiene il testo dell'edizione interpretativa del Vercelli Book, entrambi i file vengono creati durante le trasformazioni XSLT.

Per agevolare la ricerca delle parole sul Vercelli Book è stata inserita una tastiera virtuale contenente caratteri non disponibili immediatamente su una tastiera fisica del computer.

- **Evidenziatore di parole chiave**: l'evidenziazione di parole chiave all'interno dei testi è stata realizzata tramite plug-in che utilizzano Javascript e la manipolazione DOM, utilizzando poi CSS per modificare lo stile e far risaltare le parole evidenziate.
  Il testo all'interno di EVT è rappresentato come una combinazione di testo ed elementi `<span>`[52] al cui interno troviamo informazioni sul funzionamento e su come il testo deve essere visualizzato, questo tipo di markup creava problemi al plug-in originale in quanto molte parole erano spezzate dai tag e quindi non comparivano durante le ricerche; per risolvere il problema è necessario un nuovo algoritmo più sensibile nella ricerca del contenuto, l'algoritmo deve tener traccia di tutti gli `<span>` e dei nodi di testo, dove iniziano e dove finiscono.
  Al suo stato attuale EVT è in grado solamente di visualizzare un'edizione in formato diplomatico[53] e interpretativo[54] un obiettivo futuro è quello di offrire un supporto per il modulo critico TEI, per poter supportare un apparato critico[55], per poter integrare il modulo critico è necessario

---

[52] È un contenitore generico che può essere annidato. È un elemento in linea, cioè non va a capo e continua sulla stessa linea del tag che lo include.
http://www.w3schools.com/tags/tag_span.asp

[53] L'edizione diplomatica di un testo è la riproduzione fedele in caratteri di stampa (compresi errori di stampa, abbreviazioni, lettere diverse o usi non corrispondenti a quello moderno). Il nome deriva dalla disciplina diplomatica in cui è fondamentale replicare con assoluta fedeltà i documenti giuridici. (Wikipedia, voce edizione).

[54] L'edizione interpretativa è la riproduzione del testo in caratteri a stampa, ma, a differenza della precedente, lo adatta all'uso, e quindi interpreta i segni grafici per dar loro coerenza linguistica: unisce o separa le parole, scioglie le abbreviazioni, aggiunge apostrofi e accenti, rivede la paragrafatura e segnala gli errori e le lezioni sospette. (Wikipedia, voce edizione).

[55] L' edizione critica ricostruisce il testo presumibilmente voluto dall'autore, confrontando

modificare la grafica di EVT in quanto va gestito il modo in cui viene visualizzata la connessione tra il testo e l'apparato critico, ma come abbiamo visto cambiare la grafica non è complesso, il problema principale all'implementazione di un formato critico è la scalabilità del sistema (dovuto alla gestione di un possibile grande numero di commenti al testo), oltre al modulo TEI che non soddisfa a pieno tutte le necessità di EVT e deve essere migliorato e approvato dal consorzio.

Queste difficoltà rendono l'implementazione dell'apparato critico un progetto a lungo termine.

Oltre ai possibili cambiamenti che interessano il testo e le ricerche su di essi, EVT vuole migliorare anche la manipolazione sulle immagini avvalendosi di Digital Lightbox [56] un software che offre una ricca serie di strumenti: oltre alle funzioni base, come il ridimensionamento e la rotazione dell'immagine Digital Lightbox offre una buona serie di filtri, ad esempio possono essere modificate la luminosità, l'opacità, il contrasto, l'inversione di colore e la rappresentazione dell'immagine in scala di grigio, questi filtri utilizzati insieme possono migliorare notevolmente la qualità delle immagini digitali.

Digital Lightbox permette inoltre il confronto di più immagini restituendo in maniera automatica una terza immagine formata dalle differenze delle immagini messe a confronto.

La particolarità più importante di questo software che lo differenzia dagli altri editor di immagini è la possibilità di condividere il lavoro svolto con altre persone, abbiamo la possibilità di creare raccolte di immagini ed esportarle in formato XML; questa caratteristica incentiva molto la collaborazione fra utenti ed è proprio quello che si vuole raggiungere data la natura didattica di EVT.

L'implementazione completa di Digital Lightbox e di tutte le sue funzioni necessita di un architettura lato server per questa e altre necessità tra le quali inserire un database XML più potente e poter sfruttare tutte le caratteristiche di

---

tra loro tutti i manoscritti che trasmettono quel testo. (Wikipedia, voce edizione).

[56] Digital Lightbox è stato sviluppato all'origine dall'Università di Pisa, e in seguito dal College King di Londra come parte del progetto DigiPal; è un visualizzatore di immagini basato sul web realizzato per sostenere storici, paleografici e studi sull'analisi di oggetti con valore culturale. Digital Lightbox è stato sviluppato utilizzando alcune delle ultime tecnologie disponibili per il web come HTML5. CSS3 e Javascript.
https://github.com/Gbuomprisco/Digital-Lightbox/

Tipue Search, EVT potrebbe lasciare l'architettura lato client per passare ad una più complessa lato server.

Se verrà deciso questo cambio di rotta il passaggio dovrà garantire la stessa semplicità e flessibilità dalle versione odierna.[57]

Conclusioni

Iniziato come un progetto sperimentale, basato sull'esplorare questioni relative alle pubblicazioni di edizioni digitali codificate con TEI, il software è cresciuto fino a diventare uno strumento di grande utilità per l'intera comunità, poiché richiede poca configurazione, e nessuna conoscenza di linguaggi di programmazione ad eccezione di ciò che è necessario per applicare un foglio di stile XSLT.

La sua architettura lato client rende molto facile il processo di creazione, l'effettuare test sull'edizione (basta eliminare le cartelle di output e ricominciare da capo) e il pubblicare versioni preliminari sul web (tramite un qualsiasi servizio di cloud).

La fase di sviluppo per il rilascio nel 2013 del Vercelli Book in formato digitale ha creato delle ottime basi su cui costruire poi nuove funzionalità, e migliorare quelle esistenti.

Il futuro pone gli sviluppatori davanti a diverse scelte:

- Sviluppare o meno un supporto per le edizioni critiche.
- Come modificare la grafica per permettere un integrazione ottimale delle nuove funzionalità.
- Decidere se passare a un architettura lato server per poter usufruire a pieno delle funzionalità offerte da Digital Lightbox e Tipue Search.

Al momento non è chiaro quale strada prenderà EVT ma sicuramente continuerà ad essere uno strumento importante a disposizione dell'intera comunità accademica.[58]

---

[57] Cfr. Roberto Rosselli del Turco. 2013. Op. cit. http://jtei.revues.org/1077 (Visitato il 12 Marzo 2015).

[58] Cfr. Roberto Rosselli del Turco. 2013. Op. cit. http://jtei.revues.org/1077 (Visitato il 12 Marzo 2015).

Capitolo 4

HTML 5

Un nuovo standard

EVT come abbiamo visto utilizza nella sua architettura alcune delle ultime tecnologie, vediamo nello specifico cos'è HTML5.

HTML5 rappresenta la quinta versione del linguaggio su cui si basano tutte le pagine internet, ovvero HTML (Hyper Text Markup Language); è l'ultima versione di un linguaggio che si è evoluto nel corso del tempo per adattarsi alle nuove esigenze di comunicazione e pubblicazione all'interno di Internet.

Il 28 Ottobre del 2014 W3C[59] rilascia le specifiche complete[60] su HTML5, allo scopo di delineare le caratteristiche della nuova versione del linguaggio; il team che ha lavorato alla stesura di HTML5 si è posto due obbiettivi primari il primo è proporre una serie di nuove caratteristiche in grado di ampliare le possibilità offerte da HTML4, il quale non riesce più a stare al passo con le tecnologie e gli scenari moderni; l'idea alla base è quella di renderlo un linguaggio autonomo, eliminando per esempio le estensioni esterne, come Adobe Flash, per fornire le stesse funzionalità direttamente all'interno del codice (grazie agli elementi `video` e `audio`), eliminando così problemi di portabilità dei siti e dei loro contenuti.

Il secondo obbiettivo è cercare di garantire una maggiore compatibilità tra i diversi browser, indipendentemente dalle piattaforme software utilizzate e dalle differenti tipologie di dispositivi presenti sul mercato, in particolare quelli mobili (smartphone e tablet); la strada che ha intrapreso HTML5 per risolvere questo problema è favorire la scissione tra la struttura delle pagine, definita dai markup, e la loro rappresentazione, gestita tramite gli stili CSS.

---

[59] W3C (World Wide Web Consortium) è un organizzazione internazionale che sovraintende alla definizione delle specifiche, dei protocolli e delle linee guida che riguardano il World Wide Web. Nato nell'ottobre del 1994, oggi conta tra le sue fila quasi 400 membri provenienti da tutto il mondo. http://www.w3.org/

[60] Le specifiche complete si possono trovare all'indirizzo: http://www.w3.org/TR/2014/REC-html5-20141028/.

Questa scissione garantisce una maggiore standardizzazione nella visualizzazione della pagine da parte dei browser.[61]

Rispetto ad HTML4, HTML5 porta molte novità, che permettono un approccio migliore nella strutturazione dei contenuti delle pagine:

- Le regole sulla struttura del testo si sono fatte più rigide: vengono aggiunti una serie di tag semantici (ad esempio, `article`, `section`, `header` e `footer`), orientati a descrivere in modo preciso i contenuti in base al loro significato. Parallelamente all'aggiunta di alcuni tag vengono eliminati quelli obsoleti o di scarso interesse, e vengono estesi a tutti gli elementi del markup alcuni attributi, principalmente finalizzati all'accessibilità, finora previsti solo per un numero limitato di tag.
- Gli elementi per la gestione delle form vengono aumentati, sia per migliorare l'interazione utente durante l'inserimento di dati, sia per supportare maggiormente i diversi browser e dispositivi presenti sul mercato.
- Introduzione delle tecnologie SVG[62] e `canvas`[63], permettono l'utilizzo di Javascript per creare animazioni e grafica vettoriale.
- Introduzione dei tag `video` e `audio`; grazie alle quali non c'è più la necessita di ricorrere a tecnologie proprietarie per visualizzare contenuti multimediali.[64]

HTML5 non si limita a migliorare le caratteristiche proprie del linguaggio ma abbraccia anche le novità riguardanti i fogli di stile CSS (Cascading Style Sheets), grazie all'avvento di CSS3, e il linguaggio Javascript.

CSS3 include una vasta gamma di stili ed effetti applicabili direttamente ai contenuti senza dover adattare il markup, rendendo possibile una personalizzazione grafica senza precedenti; possiamo applicare gli stili agli

---

61 Cfr. Bochicchio, Casati, Civera, Golia, Mostarda *HTML5 con CSS3 e ECMAScrippt5.* Milano. Ulrico Hoepli pp.3-6

62 SVG (Scalable Vector Graphics) è un formato di immagine vettoriale basato su XML per la grafica bidimensionale con supporto per l'interattività e l'animazione. La specifica SVG è uno standard aperto sviluppato dal W3C dal 1999. (Wikipedia, voce scalable vector graphics). http://www.w3.org/Graphics/SVG/

63 `canvas` è un nuovo elemento di HTML5 il cui intento è rispondere all'esigenza di avere una superficie dove renderizzare in maniera immediata bitmap. (Wikipedia, voce canvas) http://www.w3.org/TR/html-markup/canvas.html

64 Cfr. Bochicchio, Casati, Civera, Golia, Mostarda Op. cit. pp. 12

elementi del DOM utilizzando query di selezione complesse, per esempio, in base alla posizione o alla gerarchia, possiamo applicare trasparenze, gradienti, ombreggiature, immagini di sfondo multiple, creare bordi arrotondati e utilizzare font personalizzati, il che permette di gestire gli effetti grafici più comuni senza dover ricorrere a immagini aggiuntive.

Parallelamente a CSS3 anche Javascript presenta non poche novità; con HTML5 vengono introdotte nuove API (Application Programming Interface) delle librerie Javascript che hanno lo scopo di colmare molte delle lacune presenti fino ad oggi nel linguaggio di programmazione.

Le API Javascript e le loro funzionalità riguardano:[65]

- L' accesso agli elementi DOM tramite ricerche basate sui selettori CSS3, queste API sono dette Selectors API[66].
- Il supporto alla geo localizzazione, per determinare la posizione geografica del client. Si tratta di una funzionalità motivata principalmente dalla diffusione dei dispositivi mobili dotati di GPS; queste API sono dette Geolocation API[67].
- Un sistema di memorizzazione alternativo ai normali cookie, in grado di consentire la persistenza dei dati sul client (session storage e local storage), con un notevole risparmio di banda nella comunicazione col server; queste API sono dette Web Storage API e le analizzeremo nel dettaglio più avanti.
- Possibilità di utilizzare le applicazioni web anche in modalità offline (Application Cache[68]) e di gestire l'esecuzione del codice in parti separate, garantendo un certo livello di isolamento (Web Workers[69]).
- Possibilità di modificare la cronologia di navigazione del browser; queste API sono dette History API[70].

---

65 Cfr. Bochicchio, Casati, Civera, Golia, Mostarda Op. cit. pp. 13-14

66 Le specifiche complete sono disponibili all'indirizzo http://www.w3.org/TR/selectors-api/.

67 Le specifiche complete sono disponibili all'indirizzo http://dev.w3.org/geo/api/spec-source.html.

68 Le specifiche complete sono disponibili all'indirizzo http://www.w3schools.com/html/html5_app_cache.asp.

69 Le specifiche complete sono disponibili all'indirizzo http://www.w3schools.com/html/html5_webworkers.asp.

70 Le specifiche complete sono disponibili all'indirizzo

- Lo scambio online di informazioni tra diversi domini; queste API sono dette Messaging API[71].

HTML5 non si occupa quindi solamente del linguaggio di markup ma include al suo interno un insieme di tecnologie tra loro complementari e caratterizzate da finalità diverse, ciascuna delle quali fornisce soluzioni ai diversi problemi che possiamo incontrare durante la realizzazione delle pagine web.

Il supporto ad HTML5 da parte dei browser

Nonostante HTML5 sia di fatto il nuovo standard è ancora molto giovane, e questo porta a non avere un supporto totale da parte dei browser; che di fatto non supportano alcune delle nuove funzioni messe a disposizione.
Per avere un idea precisa sul tipo di supporto offerto ad HTML5 dal browser che stiamo utilizzando possiamo effettuare un test sul sito html5test che, pur non essendo uno strumento ufficiale e riconosciuto dal consorzio W3C, offre delle indicazioni precise suddividendo le caratteristiche di HTML5 in 8 insiemi: tag semantici, multimedia, 3d grafica ed effetti, accesso ai dispositivi, offline e immagazzinamento, performance e integrazione, connettività, altro.
L'insieme di queste caratteristiche classificano i browser, i quali possono ottenere un punteggio che va da un minimo di 0 ad un massimo di 555 in base al supporto offerto alle novità di HTML5.
Ad oggi il browser con il punteggio più alto è Google Chrome che totalizza 505 punti, mentre in coda troviamo Internet Explorer 11 con 343 punti.
C'è da sottolineare che molte delle caratteristiche non supportate sono delle funzionalità molto particolari, e ad ogni rilascio le compagnie cercano di implementare sempre più funzioni, ma comunque i risultati ottenuti dai test confermano come il linguaggio HTML5 sia ancora molto giovane e probabilmente servirà ancora tempo prima di avere una situazione realmente stabile, che garantisca una compatibilità totale attraverso tutti i tipi di browser e di piattaforme.

---

http://www.w3.org/TR/2011/WD-html5-20110113/history.html.
Non sono complete e possono contenere bug.

[71] Le specifiche complete sono disponibili all'indirizzo http://www.w3.org/TR/messaging-api/.

Web Data Storage

Il professor Del Turco mi ha chiesto di rendere permanente la scelta della lingua selezionata, così da non doverla cambiare ogni volta che EVT viene caricato. Non potendo usufruire di un server per salvare i dati, è nato il problema di come preservare le informazioni nel tempo.
In HTML4 l'unico modo per poter avere dei dati persistenti utilizzando un architettura lato client è quello di utilizzare i cookie, ma il loro utilizzo ha diversi svantaggi:[72]

- I cookie sono inclusi ad ogni richiesta HTTP, rallentando di conseguenza l'applicazione web, questo perché gli stessi dati vengono trasmessi più volte.
- I cookie essendo inclusi ad ogni richiesta HHTP, inviano dati non crittografati su Internet, rendendo i dati e la privacy vulnerabili.
- I cookie sono limitati a circa 4 kb di dati, sufficienti a rallentare l'applicazione, ma non ad offrire uno spazio sufficientemente grande.

Quello di cui avevo bisogno era un modo alternativo che limitasse o eliminasse del tutto i problemi dati dai cookie; la soluzione è stata trovata grazie ad una nuova funzione di HTML5, le web data storage.
Con le storage API c'è la possibilità di gestire una grande mole di dati, direttamente all'interno de browser, generalmente il limite è impostato a 5 Mb (contro i 4 kb dei cookie) per ogni applicazione, e al contrario dei cookie i dati non vengono mai trasmessi a un server ma vengono immagazzinati direttamente all'interno del browser web.
HTML5 dispone di due diversi tipi di archivi, `local storage` e `session storage` l'unica differenza tra i due archivi è che il primo risulta persistente a tempo indeterminato mentre il secondo viene automaticamente svuotato una volta terminata la sessione di navigazione.
Tutte e due gli archivi sono formati da insiemi di coppie di valore, gestibili come un array associativo o attraverso i seguenti metodi:

---

[72] Cfr. Mark Pilgrim. *The Past, Present & Future of Local Storage for Web Application.* 2009 http://diveintohtml5.info/storage.html (Visitato il 07 Aprile 2015).

- `length`: determina il numero totale degli elementi memorizzati.
- `key(indice)`: serve per ottenere una chiave in base alla sua posizione nell'indice.
- `getItem(chiave)`: serve per ottenere un valore memorizzato.
- `setItem(chiave, valore)`: serve per salvare un nuovo elemento, fornendo la chiave e il relativo valore associato.
- `removeItem(chiave)`: serve per eliminare un elemento.
- `clear()`: serve per rimuovere tutti gli elementi precedentemente memorizzati.[73]

Tutte le ultime versioni dei browser supportano questa funzione, ma è sempre meglio stabilire se il browser in uso supporta effettivamente le API web data storage.
Per determinare se l'archivio (persistente o di sessione) richiesto è disponibile si utilizza una semplice funzione:[74]

```
function Disponibile(StorageInSessione) {
var disp
StorageInSessione?[75] "localStorage":"sessionStorage";
return(SeStorageDisponibile(disp))? windows[disp]: null;
};
```

Le modalità di accesso all'archivio locale e a quello di sessione sono identiche perché entrambi utilizzano la stessa interfaccia; i dati (sia le chiavi che i valori) vengono di norma salvati nell'archivio come stringhe (`DOMString`, nonostante la specifica ammetta qualsiasi tipo), per cui potrebbe risultare necessario effettuare delle conversioni in fase di estrazione; un modo per risolvere questo problema è serializzare i valori salvati nell'archivio in formato JSON (Javascript Object Notation), in quanto, oltre a semplificare la conversione dei tipi primitivi

---

73 Cfr. Bochicchio, Casati, Civera, Golia, Mostarda Op. cit. pp. 242.
74 Cfr. Bochicchio, Casati, Civera, Golia, Mostarda Op. cit. pp. 243.
75 Condizione ? true (Vero) : false (Falso).
La condizione si dovrà verificare ? (Equivalente di un if) in questo modo se è vera, : (equivalente di un else) oppure in quest'altro se è falsa.

come valori booleani, numeri e date, consente anche di gestire facilmente tipi complessi come oggetti, array etc.

Se il quantitativo di dati eccede la dimensione massima consentita dalle impostazioni (di default 5 Mb ma è possibile cambiarlo dalle impostazioni del browser), viene generato un errore: `QUOTA_EXCEEDED_ERR`, e l'operazione di scrittura non viene eseguita.[76]

Chiamare la funzione `setItem()` con il nome di una chiave già esistente fa sì che il dato precedente venga sovrascritto; mentre chiamare una funzione `getItem()` con una chiave inesistente restituisce `null`.

La sintassi per immagazzinare un dato è dunque questa:

```
localStorage.setItem("chiave","valore");
```

Mentre la sintassi per recuperare il dato è la seguente:

```
document.getElementById("risultato").innerHTML = localStorage.getItem("chiave");
```

Viene creato un localStorage con la coppia chiave/valore (nell'esempio la chiave ha valore "chiave" e il valore "valore"), il valore viene poi recuperato e inserito all'interno dell'elemento con id=" risultato"; se volessimo togliere questo dato dall'archivio utilizzeremo il metodo `removeItem()`:

```
localStorage.removeItem("chiave");
```

[76] Cfr. Mark Pilgrim. Op. cit. http://diveintohtml5.info/storage.html (Visitato il 07 Aprile 2015).

Capitolo 5

Creazione di un EVT Multilingua

Introduzione e spiegazione del plug-in

Il professor Roberto Rosselli Del Turco mi ha chiesto di realizzare una versione multilingua di EVT, per poterne aumentare l'accessibilità e l'usabilità.
Il codice che permette la visualizzazione di EVT in diverse lingue (inglese, francese, italiano) è inserito nella cartella plug-in la quale al suo interno contiene tutti i codici esterni di EVT, il nome del mio plug-in è **jquery-lang.js**.
Non potendo lavorare lato server, vista l'architettura totalmente lato client di EVT, il codice è stato scritto in Javascript, e per poter rendere la scelta dell'utente persistente ci si affida alle funzionalità messe a disposizione dalle API Storage.
La manipolazione dei dati è raggiunta in maniera più semplice grazie all'utilizzo di alcuni metodi messi a disposizione da jQuery:

- `.each`: permette di scorrere una qualsiasi raccolta (oggetto o matrice)
- `.attr`: il metodo restituisce il valore di un dato attributo.
- `.is`: esegue un test booleano sull'elemento selezionato utilizzando un selettore come argomento.
- `.val`: questo metodo è stato creato per avere accesso al contenuto di tutti gli elementi che caratterizzano una form HTML; è inoltre possibile stabilire il valore che un elemento deve assumere.
- `.data`: permette di allegare qualsiasi tipo di dato a un elemento DOM in modo sicuro senza rischi di perdita di memoria.
- `.html`: permette di ottenere il contenuto di qualsiasi elemento HTML.

Il codice lavora sul tag `lang` HTML e in base a come viene modificato cambia le parole associate a quell'attributo; la lingua preselezionata è l'inglese e può essere modificata facendo click sulle bandierine.
All'attivazione di una bandiera il tag `lang` viene modificato in base alla lingua selezionata, e una volta modificato l'attributo `lang` viene richiamata la funzione:

```
17 <a href="javascript:void(0);" onclick="window.lang.change('en');">
18 <img src="images/en.gif" class="flag" lang="en" title="English"
19 alt="ENG"></a>
```

Il plug-in essendo lato client non ha la possibilità di accedere a un traduttore online, viene quindi creato un vocabolario e tramite Javascript le parole vengono sostituite in base alla lingua scelta.

Per poter effettuare questa sostituzione è necessario creare un file per ogni lingua che si vuole rendere disponibile; i file sono inseriti all'interno della cartella langpack e vengono chiamati come il tag `lang` a cui appartengono, nella mia cartella troviamo i file **fr.js** e **it.js.**

Questi file Javascript contengono l'elenco di tutte le parole o frasi presenti sulla pagina, che possiedano l'attributo `lang,` e la rispettiva traduzione nella lingua desiderata.

Dall'inglese al francese:

```
1 jquery_lang_js.prototype.lang.fr = {
2     'Folio':'Folio',
3     'TextLink':'TextLink',
4     'Edition':'Édition',
5     'Text':'Texte',
6     'Search':'Recherche',
7     'Next':'Suivant',
8     'Previous':'Précédent',
9     'Expand frame':'Élargis le cadre',
10    'Thumbnails':'Miniatures',
```

Dall'inglese all'italiano

```
1 jquery_lang_js.prototype.lang.it = {
2     'Folio':'Foglio',
3     'TextLink':'Textlink',
4     'Edition':'Edizione',
5     'Text':'Testo',
6     'Search':'Ricerca',
7     'Next':'Avanti',
8     'Previous':'Indietro',
9     'Expand frame':'Espandi',
10    'Thumbnails':'Miniature',
```

Quando cambiamo lingua il codice effettua una serie di `if` a cascata con lo scopo di controllare tutti gli elementi HTML e ricercare l'attributo `lang` all'interno di essi; poi controlla se la parola/frase all'interno dell'attributo è presente nei file vocabolario, in caso positivo effettua la sostituzione.
Il codice per la trasformazione è il seguente:

```
123  var newText = this.lang[lang][defaultLangText] || currentText;
124  var newHtml = currentText.replace(currentText, newText);
125  langElem.attr('title', newHtml);
126  if (currentText != newHtml) {
127      langElem.attr('lang', lang);
128     }
```

Come detto la lingua base è l'inglese ma grazie al `localstorage` il cambio non si annullerà dopo la chiusura del browser.

```
238  jquery_lang_js.prototype.update = function (lang) {
239      if (localStorage) {
240          localStorage.setItem('langJs_currentLang', lang);
241      }
242      this.emit('update', lang);
243  }
244
```

Questo estratto di codice salva la lingua selezionata.
Quando la pagina viene aperta viene controllato se all'interno del `localstorage` sono salvati dei dati, in caso positivo si modifica in automatico la lingua, mentre in caso negativo viene mantenuto l'inglese.

```
89       if (localStorage) {
90           var lsLang = localStorage.getItem('langJs_currentLang');
91           if (lsLang) {
92               this.change(lsLang);
93           }
94           else
95               this.change(this.currentLang);
96       }
97   }
```

Il codice per poter funzionare correttamente fa uso di alcune nuove funzionalità messe a disposizione da HTML5; in particolar modo sfruttano le API Storage in

modalità persistente (`localstorage`), tralasciando la modalità sessione (`sessionstorage`).

Integrazione del codice su EVT

Come detto nel capitolo 3 EVT è formato da una struttura modulare, che permette di integrare nuove funzioni al suo interno senza andare a modificare la struttura generale del software.

Per integrare il mio lavoro è stata necessario modificare diversi file; da un lato bisogna gestire la trasformazione XSLT e integrare la mia parte all'interno delle trasformazioni già esistenti, dall'altro lato va gestita l'interazione dell'utente tramite Javascript.

Questi passaggi sono stati realizzati seguendo il metodo standard utilizzato dal team di EVT.

Il codice HTML è stato isolato all'interno di in un file separato, **tabellaprova.xsl**, in modo da mantenere il codice modulare, il file poi è stato linkato all'interno di **evt_builder.xsl**, file che dà il via alla catena di trasformazioni XSLT

Il collegamento è stato fatto tramite questa stringa di codice:

```
61  <xsl:include href="modules/fundamental_units/tabellaprova.xsl"/>
```

Una volta effettuato il collegamento è stata creata una variabile nel file di configurazione **evt_builder-conf.xsl**, file al cui interno troviamo tutte le variabili per l'output html.

In questo caso c'era bisogno di un valore booleano:

```
51  <xsl:param name="tabellaProva" select="true()"/>
```

Per richiamare il template in fase di trasformazione, è stato aggiunto un comando nel file **evt_builder-main.xsl**:

```
71 <xsl:if test="$tabellaProva=true()">
72     <xsl:result-document method="html" encoding="UTF-8"
73     media-type="text/plain" byte-order-mark="yes"
74     href="{$filePrefix}/data/output_data/header/tabellaInfo.html"
75     indent="yes">
76         <xsl:call-template name="tabella_generation">
77         </xsl:call-template>
78     </xsl:result-document>
79 </xsl:if>
```

Il codice effettua un controllo sulla variabile booleana precedentemente impostata nel file di configurazione; nel caso abbia valore `true`, si procede alla creazione del file **tabellaInfo.html** il quale contiene l'output delle trasformazioni generate dal template chiamato "`tabella_generation`".
Il file **tabellaInfo.html** viene salvato nella cartella di output dei dati, insieme a tutti gli output generati dal sistema.
Seguendo elementi simili è stato inserito all'interno di **evt_builder-structure.xsl** il seguente codice:

```
143 <xsl:if test="$tabellaProva=true()">
144     <tabellaInfo active="1"/>
145 </xsl:if>
```

Se la variabile di configurazione è impostata su `true`, aggiunge al file **structure.xml,** l'elemento `<headerInfo active="1"/>` il quale viene poi utilizzato da Javascript per sapere se mostrare o meno gli eventi legati alla gestione dell'output prodotto dalle trasformazioni xsl.
L'ultima modifica alla configurazione di EVT è stata la preparazione dell'area vuota utilizzata per contenere l'output generato dalle trasformazioni, il file che permette la creazione di quest'area è **evt_builder.callhtml.xsl.**
Il codice per creare l'area è il seguente:

```
174 <xsl:if test="$tabellaProva=true()">
175         <div id="tabellaInfo"></div>
176 </xsl:if>
```

Creata l'area è necessario un bottone per visualizzarla e un codice Javascript per gestire l'apertura e chiusura della stessa.
Il bottone è generato in questo modo:

```
168 <xsl:if test="$tabellaProva=true()">
169     <a href= "javascript:void(0);" id="switchForm"
170     lang="en" title="Options On/Off">
171         <img src="images/option.png" class="mainHeaderimg" >
172         </img></a>
173 </xsl:if>
```

All'interno del file **evt_builder.callhtml.xsl** sono inseriti anche i percorsi al codice esterno, nel mio caso elenchiamo i percorsi ai file Javascript

```
92  <script type="text/javascript" src="{$html_path}/js/plugin/jquery-lang.js"/>
93  <script type="text/javascript" src="{$html_path}/js/plugin/langpack/it.js"/>
94  <script type="text/javascript" src="{$html_path}/js/plugin/langpack/fr.js"/>
```

Finita la parte di caricamento e configurazione del file **tabellaprova.xsl**, resta da gestire la parte Javascript per il caricamento dei codici esterni e interazione con l'utente.
Il file che gestisce tutto il codice Javascript di EVT è chiamato **interface_control.js**, e per gestire la mia parte è stato scritto il seguente codice:

```
411 if ((($(xml).find('tabellaInfo').length > 0)&&
412 ($(xml).find('tabellaInfo').attr('active')==1)){
413 $('#tabellaInfo').load
414 ("data/output_data/header/tabellaInfo.html#tabella_cont", function(){
415 });
```

Questo codice va a leggere il contenuto del file **tabellaInfo.html**, cerca l'elemento con id `tabella_cont`, e carica il suo contenuto all'interno dell'elemento `tabellaInfo` che è presente nel file **index.html** (file che raggruppa al suo interno i risultati delle varie trasformazioni XSLT).
Il codice Javascript che permette di far funzionare il cambio lingua è situato fuori dall'elemento `tabella_cont`, è necessaria quindi una funzione che venga lanciata quando il contenuto di `tabella_cont` viene caricato completamente in `tabellaInfo`.

Per svolgere questo compito in Javascript esistono delle funzioni chiamate callback le quali partono alla fine di altre.

```
window.lang = new jquery_lang_js();
      $().ready(function () {
      window.lang.run();
     });
```

Gestito il caricamento del codice esterno, resta il codice che permette l'interazione con l'utente, per aprire e chiudere il contenitore.

```
415 $('#switchForm').click(function(){
416     if($('#tabellaInfo').is(':visible')){
417        $('#tabellaInfo').hide('drop',{direction:'down'},'linear');
418        $('#switchForm').removeClass('button_effect');
419     }else {
420         $('#tabellaInfo').show('drop',{direction:'up'},'linear');
421         $('#switchForm').addClass('button_effect');
422            }
423     });
424 }
```

Adesso il codice è completamente integrato all'interno di EVT resta solamente da gestire la parte CSS.

Il foglio di stile che gestisce EVT è il file **main.css**, nel quale sono inserite tutte le regole per la corretta visualizzazione del software, font, colori, spazi etc.

La prime regole servono per la corretta visualizzazione del div `tabellaInfo` `<div>` principale al cui interno troviamo tutti gli altri contenitori), la prima regola serve per non visualizzare il contenitore all'apertura della pagina, è poi compito del Javascript cambiare la regola quando clicchiamo il pulsante e desideriamo vedere il contenuto del div `tabellaInfo`, quando il `<div>` risulta visibile le altre due regole permettono al contenitore di stare in primo piano.

```
1490 #tabellaInfo{
1491   display:none;
1492   z-index:99999;
1493   position:absolute;
1494 }
```

Il pulsante delle opzioni per essere integrato correttamente deve avere gli stessi effetti grafici dei pulsanti già presenti in EVT, deve quindi illuminarsi al passaggio del mouse e quando `tabellaInfo` è visibile, mentre se il contenitore delle opzioni non è visibile il pulsante dovrà mostrarsi spento; per fare questo quando il pulsante è attivo il codice Javascript aggiunge al div `switchForm` la classe `button_effect`, che gestisce gli effetti grafici sul pulsante.

```
.mainHeaderimg:hover, .button_effect .mainHeaderimg {
    opacity: 1;
}
```

Tutte le altre regole servono per adattare il contenitore e il pulsante allo stile generale di EVT.

```
#tabella_cont img{
    width:24px;
    border: 1px solid #4E443C;
    margin-top:13px;
    margin-right:3px;
    margin-left:3px;
    float:right;
    }
#switchForm  .mainHeaderimg {
    width:29px;
    margin-bottom:2px;
    margin-left:10px;
}
#switchForm  .a {
    width:29px;
    margin-bottom:2px;
    margin-left:10px;
}
```


```
#tabella_cont img{
    width:24px;
    border: 1px solid #4E443C;
    margin-top:13px;
    margin-right:3px;
    margin-left:3px;
    float:right;
    }
#switchForm  .mainHeaderimg {
    width:29px;
    margin-bottom:2px;
    margin-left:10px;
}
#switchForm  .a {
    width:29px;
    margin-bottom:2px;
    margin-left:10px;
}
```


L'insieme di queste regole permette al codice di inserirsi in maniera adeguata all'interno di EVT, integrandosi perfettamente allo stile e alla grafica generale del software.

A questo punto EVT offre una nuova funzionalità ai suoi utenti: il multilingua.

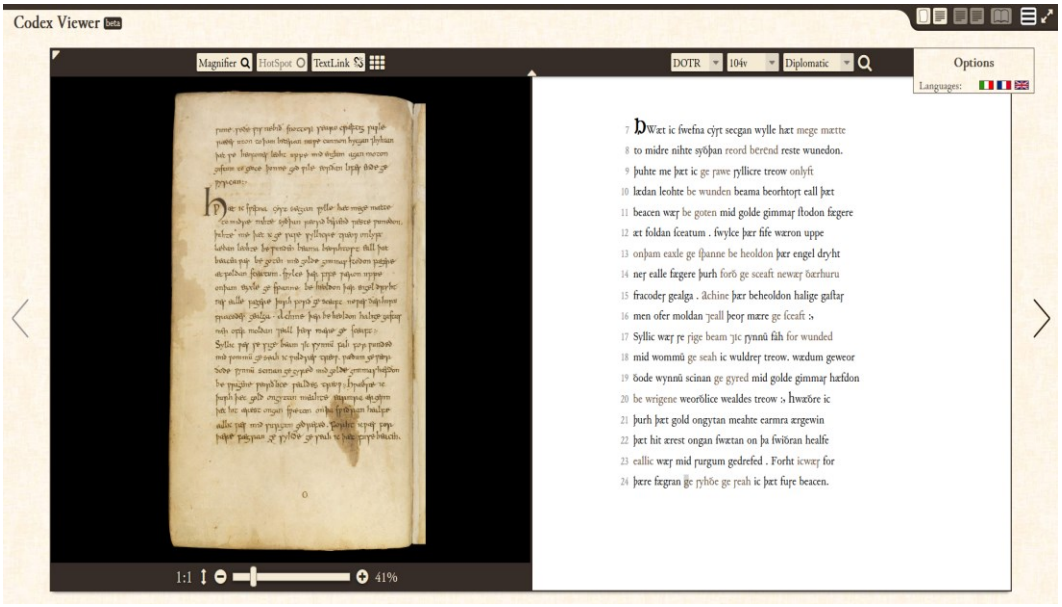

Figura 3. Interfaccia Grafica EVT.

Conclusioni

La creazione di un’edizione digitale come emerge da questo elaborato è un lavoro molto impegnativo che richiede una lunga fase di studio e impegno da parte dei programmatori.

Al Vercelli Book digitale e alla sua interfaccia grafica EVT il team del professor Roberto Rosselli Del Turco sta lavorando da alcuni anni, e nonostante abbia raggiunto una versione stabile e ben funzionante, il progetto non si è concluso ma cerca sempre di migliorarsi integrando nuove funzionalità e cercando di stare al passo con le opportunità offerte dalle nuove tecnologie (HTML5, CSS3, Javascript, etc.), per poter offrire uno strumento di lavoro e ricerca, a cui gli altri programmatori guardino come punto di riferimento.

Come il progetto del Vercelli Book digitale, la stessa disciplina di codifica di testi cerca di migliorarsi e stabilire degli standard per la digitalizzazione di ogni tipo di testo, cosi che le future edizioni digitali possano seguire delle precise linee guida al fine di raggiungere un testo digitalizzato che sia fruibile dall'utente in modo semplice ed efficace.

Il mio contributo al progetto EVT è stato come abbiamo visto il permettere (utilizzando strumenti di programmazione lato client) all'interfaccia grafica di poter essere visualizzata in diverse lingue.

Questo tipo di multilingua è efficace nel caso di EVT in quanto il vocabolario utilizzato è statico e non molto ampio, è quindi possibile modificare o ampliare facilmente i file al cui interno si trovano i vocabolari delle varie lingue.

L'utilizzo di HTML5 ha permesso inoltre di rendere permanente la scelta fatta dall'utente tramite le API Storage, una nuova e utilissima libreria Javascript.

Bibliografia

- Daniele Bochicchio, Casati, Civera, Golia, Mostarda. *HTML5 con CSS3 e ECMAScript5*. Milano. Ulrico Hoepli. 2011.

- Devimadesign, *Salvare recuperare dati localstorage HTML5*. 2012. http://www.devimadesign.com/corso-html5/salvare-dati-localstorage.html.

- Digisic, *Vantaggi e limiti del libro elettronico* http://www.digisic.it/documentazione/codifica_baltico/html/II.7.html.

- jQuery write less, do more. *jQuery API.* http://api.jquery.com/

- Kevin S. Kiernan, and Andrew Prescott. *Electronic Beowulf. [London]:* British Library, 1999.

- Halsall, Maureen. *Vercelli and the Vercelli Book*. *PMLA*. 84. 1969.

- Luiselli Fadda, Anna Maria. *Tradizioni manoscritte e critica del testo nel Medioevo germanico*. Roma; Bari: Laterza. 1994.

- Pilgrim Mark *Dive into html5*. 2009. http://diveintohtml5.info/.

- Richards, Mary P. *Anglo-Saxon Manuscripts: Basic Readings*. New York: Routledge, 2001.

- Rosselli Del Turco Roberto. *After the editing is done: Designing a Graphic User Interface for digital editions*. Digital Medievalist. 2011 http://digitalmedievalist.org/journal/7/rosselliDelTurco/

- Rosselli Del Turco Roberto. *EVT Development: an update (and quite a bit of history).* 2014. https://visualizationtechnology.wordpress.com/2014/01/26/evt-development-an-update-and-quite-a-bit-of-history/

- Roberto Rosselli Del Turco, Giancarlo Buomprisco, Chiara Di Pietro, Julia Kenny, Raffaele Masotti, Jacopo Pugliese. *Edition Visualization Technology: A Simple Tool to Visualize TEI-based Digital Editions.* Journal of the Text Encoding Initiative. 2013. http://jtei.revues.org/1077

- Rosselli Del Turco Roberto. *Vercelli Book Digitale.* 2015. http://vbd.humnet.unipi.it/

- Sommerville Ian. *User interface design. "Software Engineering".* 2004. http://ifs.host.cs.st-andrews.ac.uk/Books/SE7/Presentations/PDF/ch16.pdf.

- TEI. *Text Encoding Initiative.* 2013 http://www.tei-c.org/index.xml.

- W3C. *Cascading Style Sheets home page.* 2015. http://www.w3.org/Style/CSS/.

- W3C. *HTML5 A vocabulary and associated APIs for HTML and XHTML.* 2014. http://www.w3.org/TR/html5/.

- W3C. *Javascript Web APIS.* 2014. http://www.w3.org/standards/webdesign/script.html.

- W3C. *XSL Transformations (XSLT) Version 1.0.* 1999. http://www.w3.org/TR/xslt.

- W3Schools. http://www.w3schools.com/