UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Il modulo TEI Manuscript Description: gestione e visualizzazione delle informazioni nel progetto Vercelli Book Digitale.

Candidato: *Alessandro Barsi*

Relatore: *Roberto Rosselli Del Turco*

Correlatore: *Maurizio Tesconi*

Anno Accademico 2014-2015

Indice generale

Introduzione

La codifica dei testi è una disciplina che unisce tra di loro conoscenze tecniche e culturali che permettono la memorizzare di un documento su un supporto digitale allo scopo di conservarlo o utilizzarlo per elaborazioni automatiche. Sono disponibili per gli sviluppatori linee guida a cui ispirarsi per poter creare una pubblicazione digitale efficiente, che metta a disposizione dell'utente un set base di strumenti e funzionalità utili per svolgere operazioni su un testo.

In questo elaborato verranno illustrati i vari passaggi da compiere per progettare una pubblicazione digitale e verrà messa sotto esame la digitalizzazione del *Vercelli Book*, conforme alle norme di codifica TEI (Text Encoding Initiative), e l'implementazione di quest'ultimo all'interno di un'interfaccia grafica EVT (Edition Visualization Technology) creata dal team di sviluppo del Professor Roberto Rosselli Del Turco. Verrà trattata la gestione del modulo `<teiHeader>`, che contiene tutte le informazioni descrittive del testo e le informazioni sulla versione digitale che costituiranno il frontespizio elettronico della pubblicazione, e del modulo `<msDesc>`, che riporta tutte le informazioni dettagliate del manoscritto. Verranno studiati inoltre i procedimenti di integrazione e visualizzazione di queste informazioni all'interno dell'interfaccia di EVT andando a separare quelle generiche della pubblicazione (visualizzabili mediante appositi strumenti) da quelle più specifiche del manoscritto, così che, una volta inseriti diversi manoscritti all'interno della pubblicazione, sarà possibile visualizzare solo le informazioni inerenti ad un determinato manoscritto.

La mia scelta è ricaduta su questo argomento perché ho trovato interessante l'idea di lavorare in gruppo allo sviluppo di un progetto come quello del *Vercelli Book Digitale* (VBD). Il progetto dava la possibilità di approfondire l'utilizzo di XSLT che aggiunto a XML, HTML e CSS, andava a comporre un insieme di linguaggi che durante tutto il corso di laurea sono stati per me i più interessanti. Inoltre mi è sembrato che questa materia fosse il perfetto esempio del termine "Informatica Umanistica" dato che la codifica (parte informatica) ha a che fare con un testo antico (contenuto umanistico) di grande valore come il *Vercelli Book*.

L'elaborato da me scritto è strutturato in cinque capitoli. Nel primo capitolo si fa una descrizione del *Vercelli Book* per poi passare ad analizzare gli eventi storico

religiosi che hanno portato il libro in Italia. Il secondo capitolo introduce la "digitalizzazione" facendo una panoramica sul significato del termine "testo digitalizzato" e le caratteristiche e requisiti che una pubblicazione digitale deve rispettare. Nel terzo capitolo invece viene introdotto il progetto del *Vercelli Book Digitale*, le sue origini e il suo sviluppo; si descrive inoltre l'interfaccia grafica EVT e le funzionalità e strumenti implementati all'interno di essa riportando anche gli sviluppi futuri che sono ad oggi in fase di progettazione. Nel capitolo quattro si introducono le linee guida TEI e il linguaggio di programmazione XSLT, entrambi indispensabili per la realizzazione del progetto VBD. Nel quinto e ultimo capitolo invece passerò in rassegna quella che è la mia parte di progetto vera e propria: riporterò e spiegherò il codice da me sviluppato, il perché ho scelto determinate soluzioni e le problematiche affrontate durante il lavoro; il tutto è integrato con esempi, parti di codice e immagini per rendere più semplice la comprensione da parte del lettore.

Capitolo 1

1. Il Codex Vercellensis

1.1 Cos'è il Vercelli Book

Il *Codex Vercellensis*, o come meglio conosciuto nel mondo *Vercelli Book*, è un manoscritto creato verso la fine del X secolo che contiene al suo interno un insieme di opere a carattere religioso scritte in prosa e in versi, più precisamente 23 omelie[1] in prosa e 6 componimenti poetici con una metrica allitterativa[2] tipicamente anglosassone.

Scritto in inglese antico, l'antica forma dell'odierno inglese, è conservato presso la Biblioteca Capitolare di Vercelli.

1.2 Struttura e contenuto

Il *Vercelli Book* è composto da 136 fogli della dimensione di 31x20 cm in pergamena e su ogni pagina sono scritte dalle 23 alle 32 righe[3]. I fogli sono assemblati in successione creando così un fascicolo, ovvero più fogli incorporati gli uni negli altri. Questi fogli non sono tutti ben conservati, il primo foglio è del tutto illeggibile a causa nei numerosi danneggiamenti subiti nel tempo, e la maggior parte dei restati presenta dei danni causati da non si sa quale reagente chimico; inoltre non è presente la copertina che non è mai stata trovata e non sono presenti le rubriche di apertura e chiusura del volume.

Esaminando la grafia si nota subito che il *Vercelli Book* è scritto in minuscola quadrata anglosassone, un tipo di scrittura di inizio X secolo. Il tipo di carattere è così chiamato appunto per la forma quadra delle lettere, soprattutto le lettere *a*, *d*, *e*, *q*, e per la grafia delle aste discendenti e ascendenti dei caratteri sviluppata in senso verticale[4]. All'interno dei testi sono presenti iniziali di parole miniate in forma

[1] Nella liturgia cattolica, esposizione e commento di letture della messa

[2] Ripetizione, spontanea o ricercata, di un suono o di una serie di suoni, acusticamente uguali o simili, all'inizio di due o più vocaboli successivi.

[3] Cfr. Vercelli Book Digitale. http://vbd.humnet.unipi.it/?page_id=99. (Visitato il 18 Gennaio 2015)

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

zoomofrica[5].

Per quanto riguarda il contenuto il *Codex Vercellensis* è composto, come precedentemente detto, da 23 omelie e 6 componimenti poetici allitterativi. Delle 23 omelie scritte 11 sono attestate solo all'interno del *Vercelli Book* ed è per questo motivo che il manoscritto ha un grandissimo valore storico e culturale. I sei componimenti poetici invece sono: *Andreas* (ff. 29v – 52v), poema antico che racconta la storia dell'Apostolo Andrea, il poema è incompleto. *I Fati degli Apostoli* (ff. 52v-54r) è un breve componimento che parla degli eventi chiave accaduti ai dodici Apostoli dopo l'Ascensione di Gesù. *Anima e Corpo* (f. 98) è una delle due parti con il medesimo titolo che tratta il tema del dialogo tra anima e corpo, è uno dei testi più vecchi scritti in inglese antico. *Il sogno della croce* (ff. 104v-106r) è una delle prime poesie cristiane, il narratore descrive un sogno in cui appare il crocifisso. *Elena* (ff. 121r-133v) è il più lungo poema della raccolta e ha come tema il ritrovamento della Santa Croce da parte di Sant'Elena madre dell'imperatore Costantino. Vi sono inoltre un insieme di frammenti omiletici trattanti il tema dell'inganno. L'intero corpus di testi del *Codex Vercellensis* è anonimo tranne *Elena* e *Il Fato degli Apostoli* che vengono attribuiti a Cynewulf[6], grazie alla presenza di rune raffigurate sui testi che unite compongono il nome "Cynewulf"[7].

Questa raccolta di componimenti poetici e omelie è stata creata *ad hoc* e in diverse copie scegliendo le opere fra un vasto corpus di testi e si desume, guardando la calligrafia, che lo scriba che ha composto la raccolta fosse uno solo. Analizzando la grafia si nota che lo scriba non è intervenuto nella struttura del contenuto, eseguendo una copiatura del tutto meccanica e correggendo solo eventuali suoi errori di ortografia.

[5] Raffigurazione di lettere con aspetto animale.

[6] Cynewulf (IX secolo) è stato un poeta anglosassone. È famoso per le sue composizioni religiose, ed è considerato una delle figure di maggior rilievo nella poesia cristiana in antico inglese. Le sue opere sono *L'ascensione*, *Cristo*, *Elena* e *I fati degli Apostoli*. Era noto per lasciare scritti nelle sue opere caratteri runici che rappresentavano le lettere che componevano il suo nome. (Cfr. Encyclopedia Britannica, http://www.britannica.com/EBchecked/topic/148420/Cynewulf. (Visitato il 19 Gennaio 2015)).

[7] Cfr. A. M. Luiselli Fadda, *op. cit.*, p. 49.

1.3 Origini e storia

L'ipotesi che prevale fra gli studiosi sul perché sia stato scritto il *Vercelli Book* ci dice che il committente intendesse creare un florilegio[8] spirituale che fosse utile alla meditazione e alla preghiera.

Di maggiore difficoltà è invece capire il "perché" e il "come" il *Vercelli Book* sia arrivato in Italia e sia rimasto per tutti i secoli fino, ad oggi, negli archivi eusebiani di Vercelli. Diversi studiosi hanno cercato di dare una risposta a questa domanda proponendo diverse teorie, alcune molto improbabili come quelle dell'archeologo italiano Costanzo Gazzera il quale teorizza l'arrivo del manoscritto a Vercelli per mano del monaco scrittore Giovanni Scoto Eriugena[9], oppure quella dello studioso Max Förster che secondo lui il *Vercelli Book* sarebbe stato portato in città da Giovanni Francesco Bonomo Vescovo di Vercelli del XVI secolo. In diversi hanno invece teorizzato che il manoscritto arrivasse dal continente europeo da località come San Gallo[10] (Svizzera) o Fulda[11] (Germania) che avevano connessioni con Vercelli. Altre teorie hanno datato l'arrivo del manoscritto in Italia in una data più indietro nel tempo, ad esempio c'è chi sostiene che il *Vercelli Book* sia arrivato in un bagaglio del Cardinale Guala Bicchieri[12] nel XIII secolo, oppure, Richard Wülker ha definito il XIII una data troppo tarda basandosi sulle stile grafico del manoscritto e ricollocandolo quindi nell'XI secolo come un dono di un pellegrino inglese fermatosi a Vercelli presso l'ospizio di Santa Brigida degli Scoti in un suo viaggio verso Roma. Una variante della teoria di Wülker l'ha esposta S.J. Herben: in accordo con Wülker sulla data di importazione del manoscritto, Herben sostiene però che il *Vercelli Book* sia arrivato a Vercelli nel 1050 tramite Wulfwig (detto Ulf) Vescovo di Dorchester[13]. In quell'anno infatti a Vercelli si tenne un sinodo[14] dove venne

[8] Scelta di opere o di brani di opere di uno o più scrittori, raccolta in volume

[9] Monaco, teologo, filosofo e traduttore irlandese, considerato uno dei più grandi filosofi altomedievali.

[10] Città della Svizzera, le sue origini risalgono al 612 quando l'irlandese San Gallo vi costruì un eremo e da lì poi nel corso dei secoli si sviluppò la città fino ai giorni nostri.

[11] Situata nell'est dell'attuale regione dell'Assia lungo il fiume Fulda fu fondata nel 744 da San Sturm dove eresse un monastero benedettino. Nel 1752 divenne sede vescovile.

[12] Originario di Vercelli (Vercelli, 1150 – Roma, 1227), dove nel 1219 vi fondò la basilica di Sant'Andrea, fu cardinale e diplomatico al servizio dello Stato della Chiesa.

[13] Città della contea di Dorset in Inghilterra.

[14] Assemblea dei preti e di altri fedeli di una diocesi, indetta dal vescovo.

chiamato a processo Berengario di Tours[15], in quel caso il manoscritto venne utilizzato come ricompensa da parte di Ulf a garanzia della condanna di Berengario[16].

Di tutte le teorie formulate nei decenni molte vengono invalidate anche grazie ad uno studio più attento del contenuto del manoscritto stesso e di altri documenti dell'archivio eusebiano. Un ulteriore errore commesso da alcuni studiosi che hanno fatto ipotesi sul percorso che ha portato il codice a Vercelli, è stato quello di non studiare e capire bene a fondo la storia religiosa e culturale della città che è invece molto utile per trovare una teoria più appropriata.

Proviamo quindi ad analizzare la storia di Vercelli durante il Medioevo. Vercelli agli inizi della religione cristiana fu un'importante centro culturale e religioso in cui venne eretta una chiesa in onore di Sant'Eusebio di Vercelli[17] attorno alla quale sorsero molti *scriptoria* e scuole fiorenti. Vercelli essendo situata lungo il tragitto di numerose strade che dalle Alpi portavano a Roma, fu soggetta a molte incursioni barbariche (come quella dell'899 in cui andarono distrutti molti documenti e la cattedrale stessa) ma questo comportò anche un enorme vantaggio in quanto divenne un grande centro culturale e di commercio dove una gran quantità di viaggiatori, commercianti, laici, clericali, pellegrini, provenienti soprattutto dall'Inghilterra, si incontravano arricchendo finanziariamente e culturalmente la città.

La vita religiosa della città all'epoca era supervisionata dalla chiesa di Sant'Eusebio, eccezion fatta per l'abbazia di Sant'Andrea che si rese indipendente tramite un decreto papale. Con queste informazioni possiamo dire che l'ospedale di Santa Brigida degli Scoti, menzionato nella teoria di Richard Wülker, fosse sotto il controllo degli Eusebiani fino alla fine del XIV secolo quando ripresero gli scontri tra Guelfi e Ghibellini che portarono l'ospedale ad una scarsità di provviste e mezzi di sostentamento. Questo stato di indigenza costrinse il tesoriere in carica ad inviare agenti in Inghilterra alla ricerca di fondi che vennero inviati ai canonici eusebiani sotto forma di pagamenti annuali a discapito però della cessione del comando

[15] Filoso e dialettico francese (Tours, 998 – Saint Cosme, 6 Gennaio 1088) viene imprigionato e condannato al concilio di Vercelli a causa della sua tesi in proposito della trasformazione del pane e vino in corpo e sangue di cristo.

[16] Cfr. M. Halsall, *Vercelli and the Vercelli Book, Modern language association*, (1969), p. 1545.

[17] Primo Vescovo dell'Arcidiocesi di Vercelli (Sardegna, 283 – Vercelli, 1° Agosto 371) e oggi Patrono della medesima città.

dell'ospedale e alla definitiva annessione di quest'ultimo sotto il controllo di Sant'Andrea nel 1343.

Prima di questi avvenimenti Santa Brigida fu una casa per studenti inglesi in quanto nel 1228 il Comune di Vercelli era un fiorente polo universitario dove vennero a studiare inglesi, francesi, normanni, catalani, spagnoli, tedeschi e italiani, fino al 1335 quando il declino del Comune rese Vercelli una semplice cittadina di sosta lungo la strada per Roma.

Essendo un punto di fermata obbligatorio per chi proveniva dall'Inghilterra, questo ci deve far capire di come sia molto plausibile che il *Vercelli Book* arrivi direttamente da questo paese, come lo sono arrivati negli anni molti altri presenti in città. Questo ci porta a capire che la maggior parte delle teorie sulla presenza del Codice in Italia siano solo ipotesi o siano state costruite con un numero esiguo di prove e senza un profondo e attento studio religioso e storico di Vercelli durante il Medioevo[18].

Al di là di tutte le ipotesi fatte sull'arrivo in Italia del *Vercelli Book* l'unica certezza è che vi è rimasto per tutti questi secoli, ce lo dimostrano i registri compilati nei secoli dai canonici di Sant'Eusebio ritrovati all'interno degli archivi della cattedrale. Questi registri riportano tutti gli inventari fatti nei secoli e che contengono una serie di annotazioni dove si può dedurre la presenza del Codice all'interno degli archivi: uno di questi cataloghi è quello fatto da un canonico chiamato Leone che riporta, alla riga novanta, una dicitura con scritto: «*Liber Gothicus sive Longobardus (eum legere non valeo)*» (Halsall 1969, p. 1547), il testo tra parentesi significa "non riesco a leggere" e può essere riferito proprio alla lingua con cui era scritto il *Vercelli Book*. Viene menzionato un altro inventario del 1750 redatto dall'antiquario veronese Giuseppe Bianchino che riporta la seguente dicitura: «*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), la dicitura "*sæculi X*", "X secolo" in italiano, può essere ricondotta alla data di creazione del manoscritto e lo stesso la frase "*Liber ignotæ linguæ*", "libro di lingua ignota" in italiano, ci rimanda probabilmente all'inglese antico con cui è scritto il manoscritto.

[18] Cfr. M. Halsall, *op. cit*., pp. 1546-1547.

Tra tutti i cataloghi che riportano le date degli inventari fatti nel corso dei secoli ve ne sono due molto importanti, riportati di recente alla luce dagli archivi cattedrali, fatti rispettivamente nel XV e XVIII secolo. Questi due cataloghi sono da studiare a fondo in quanto sono molto importanti per ipotizzare i tempi di arrivo del *Vercelli Book* a Vercelli[19].

Riguardo al catalogo del XV secolo sappiamo che è stato compilato dal tesoriere degli Eusebiani, da un canonico e un notaio locale nel 1426, e che tale inventario è composto da 80 fogli di carta raccolti in 8 quaderni riportanti cinque marchi che confermano la creazione di quest'ultimo nel XV secolo in Nord Italia. La parte più significativa di questo inventario risiede in una lista di contenuti numerati in cui sono riportate le locazioni dei documenti all'interno della chiesa per un totale di 90 volumi. Di tali libri non è riportato solo il contenuto ma anche la descrizione fisica dell'oggetto, il nome dello scriba che li ha creati e la data in cui è stata redatta la copia. Grazie a tutte queste informazioni è possibile identificare facilmente tutti i libri all'interno della biblioteca di Sant'Eusebio come ad esempio un libro del IX secolo che riporta diverse omelie gregoriane ed intitolato Codex CXLVIII oppure un Vangelo del IV secolo attribuito al patrono della cattedrale.

Andando ad analizzare a fondo le descrizioni dei libri presenti nel catalogo ce n'è una molto interessante. Dettagliata dal punto di vista della descrizione fisica del documento in esame, risulta invece molto vaga per quanto riguarda il contenuto del manoscritto; la descrizione riporta tali parole: «*Item liber omeliarus antiquissimus non abens principiu nec finem, et aliquantulum dequantenarus, cum asseribus aliquantunum a libro remotis, scriptus in carta*» (Halsall 1969, p. 1549). Il termine "*antiquissimus*" potrebbe essere utilizzato non tanto per la data del manoscritto quanto per l'incapacità del cataloghista di decifrare il testo; questo ci porta a pensare che tale documento sia il più antico tra tutti quelli presenti. Oltre a questa descrizione ci aiuta anche una nota aggiunta che riporta la dicitura "*Omiliarum liber ignotis idiomatis*", ovvero un libro di omelie dalla lingua ignota che potrebbe essere appunto l'inglese antico. Incrociando le informazioni della descrizione e della nota scritta possiamo dire che a quel tempo si trattava di un libro di omelie scritto pergamena, senza una rubrica di apertura e chiusura, con la copertina smarrita e scritto in una

[19] Cfr. M. Halsall, *op. cit*., pp. 1547-1548.

lingua indecifrabile, indizi che ci portano a confermare che quello fosse il *Vercelli Book*[20].

Per quanto riguarda il catalogo del XVIII secolo esistono due versioni: la prima è stata riscoperta da poco dentro gli archivi della cattedrale e la seconda è in possesso del Dott. Ernesto Gorini di Vercelli. Per quanto riguarda la prima versione, consiste in un singolo quaderno dal titolo «*Recensio Codicum Msrum qui in Tabulario Vercellensis Ecclesiæ Asservantur*» (Halsall 1969, p. 1548) composto da nove fogli di carta, non rilegati, piegati e numerati da 1 a 31. Ognuna di queste pagine riporta anche la filigrana di una copisteria biellese. La versione in possesso di Gorini è intitolata «*Recensio Codicum Centum MSS. ex is, qui in Tabulario Vercellensis Ecclesiæ Asservantur*» (Halsall 1969, p. 1548) ed è formata da 20 fogli raccolti in tre quaderni da 6 fogli ciascuno e un quaderno da 2 fogli; su ogni foglio sono presenti due marchi riportanti la figura di un prete con una verga e un cimiero riportante il nome del produttore Pietro Cappuccino. Nonostante non sia riportata nessuna data sulla creazione di questo catalogo, basandosi sul contenuto della riga 41 possiamo attestare che il catalogo sia stato redatto da Carlo Luigi del Signore nel 1778. Di nostro maggiore interesse sono però le pagine 18 e 25 della prima versione (quella riscoperta negli archivi) in cui il redattore non si è limitato semplicemente a riportare le informazioni del *Vercelli Book* ottenute al suo tempo, ma si è anche impegnato a rilasciare una dettagliata e scrupolosa analisi personale del manoscritto, dei suoi contenuti e di come potevano essere decifrati[21].

Se il "*Liber Omeliaris*" citato nel catalogo del 1426 è veramente il *Vercelli Book* allora è impossibile che quest'ultimo sia veramente stato importato da umanisti del XVI secolo ma deve essere rimasto negli archivi per molti decenni addietro, abbastanza da far perdere traccia della sua provenienza. L'ipotesi che sia arrivato con Guala Bicchieri sembra quindi essere solo una storia locale in quanto l'interesse da parte del Cardinale di un documento scritto in lingua "morta" potrebbe essere solamente una supposizione in quanto non c'è nessuna prova a favore di questa tesi mentre invece è certo che alla sua morte non erano presenti libri scritti in inglese antico nella sua libreria, e se anche li avesse posseduti, sarebbero stati donati alla sua

[20] Cfr. M. Halsall, *op. cit*., pp. 1548-1549.
[21] Cfr. M. Halsall, *op. cit*., p. 1548

fondazione di Sant'Andrea dove sarebbero rimasti fino alla dissoluzione dell'abbazia nel XIX secolo.

Ciò ci porta a concludere che la teoria che ipotizza l'arrivo del *Vercelli Book* nel XI secolo sia la più veritiera in quanto il codice deve essere rimasto per un lungo periodo di tempo in Inghilterra, tanto da riportare annotazioni in lingua inglese tra le pagine, ma non abbastanza da evitare la citazione che compare nel foglio 24v, che riporta un verso del salmo XXVI in scrittura neumatica[22] con una grafia attribuibile per certo ad una mano italiana di inizio XII secolo. In quel periodo quindi il *Vercelli Book* può essere tranquillamente arrivato a Vercelli all'interno del bagaglio di uno dei tanti viaggiatori partiti dall'Inghilterra e poi fermatosi a Santa Brigida degli Scoti, da qui poi deve essere finito negli archivi cattedrali ed esservi rimasto in disparte per tutti questi secoli a causa della sua incatalogabilità fino al XIX secolo quando venne riscoperto da parte di uno studente dell'antica lingua inglese[23].

[22] Nella notazione medievale, segno grafico che suggeriva una flessione della linea melodica e ne indicava il modo d'esecuzione.
[23] Cfr. M. Halsall, *op. cit*., pp. 1549-1550.

Capitolo 2

2. La digitalizzazione

2.1 Perché digitalizzare

«Il processo di digitalizzazione consiste nella conversione di un originale in formato digitale. L'originale in questione può avere la forma più diversa: un quadro, un filmato, un documento burocratico, un brano musicale, un intero archivio fotografico, *etc*»[24].

Quindi, perché digitalizzare? La risposta è semplice, l'edizione elettronica di un oggetto che rientra in una delle categorie elencate sopra può fornire una moltitudine di vantaggi sotto ogni punto di vista. Ad esempio qualsiasi testo scritto, immagine *etc* potrebbero essere visionati su personal computer. Tramite internet, oppure duplicandolo in diverse copie, può essere facilmente condiviso e raggiungibile da chiunque rendendolo accessibile ad un grande pubblico; questo porta ad un grande vantaggio quando si pensa alle possibili problematiche che si hanno nel reperire un documento o un libro magari presso una biblioteca o un archivio perché già in possesso da altre persone o magari per svariati motivi presenta delle restrizioni sulla consultazione da parte di uno studente. Un altro vantaggio fornito da una copia digitale è quello di non andare a toccare direttamente con mano il materiale, si pensi per esempio ad un libro antico o ad una pergamena, questo ci permette di evitare che il testo da esaminare venga toccato riducendo così la possibilità di eventuali danni o usura delle parti che sono già di per sé fragili e che possiedono un grande valore storico e culturale da preservare.

Con il passare degli anni si sono evolute le tecnologie che ad oggi mettono a disposizione delle versioni digitali una gran quantità di strumenti che danno la possibilità di svolgere particolari azioni, ad esempio le immagini ad alta qualità del documento digitale danno la possibilità di evidenziare particolari o ingrandire le immagini per andare a ricercare particolari dettagli, oppure la possibilità di effettuare ricerche dirette all'interno del testo altrimenti impossibili su carta.

[24] Cfr. Vercelli Book Digitale, *Digitalizzazione delle immagini*. http://vbd.humnet.unipi.it/?page_id=148. (Visitato il 31 Gennaio 2015).

2.2 Progettazione di una versione digitale: caratteristiche e requisiti

Accertato che una versione digitale presenta molti vantaggi rispetto ad una cartacea, dobbiamo anche considerare però le difficoltà che si presentano quando deve essere progettata. Lo scopo principale di una versione digitale è quello di inserire un documento in un contesto ipertestuale[25] così da poter navigare e interagire con una gran quantità di contenuti; dal momento però che l'informatica è una scienza recente e le pubblicazioni digitali sono un oggetto dalle grandi potenzialità, ma anche complesso da realizzare, non vi è uno standard da cui partire e perciò ad oggi si creano ancora progetti *ad hoc* a seconda del tipo di testo e supporto che vogliamo digitalizzare.

Vi sono però linee guida che è possibile seguire per la realizzazione di un'ottima pubblicazione digitale. Una edizione digitale comprende di norma un'interfaccia grafica e le scansioni in alta risoluzione del contenuto, che sia un testo scritto o un insieme di fotografie e immagini. La scansione in alta risoluzione permette di avere una copia tale e quale all'originale che offre la possibilità di evidenziare anche i più piccoli particolari. Solitamente poi le immagini sono inserite all'interno di un'interfaccia grafica che mette a disposizione strumenti e funzioni per poter esplorare e lavorare sul testo o sulle immagini.

Vi sono molti tipi e modi di sviluppare interfacce grafiche per la visualizzazione di edizioni digitali; in un primo momento sono state sviluppate (e lo sono tutt'ora) interfacce per browser web[26] dal momento che il *World Wide Web* è un ottimo strumento per la diffusione dei contenuti e di facile accessibilità. Ci sono però contenuti che non possono essere condivisi in rete, spesso per motivi di diritti d'autore, e sono state create quindi interfacce grafiche *stand alone*[27] basate sempre su browser web con la possibilità di essere utilizzate anche senza essere condivise in

[25] Insieme di documenti messi in relazione tra loro per mezzo di parole chiave. Può essere visto come una rete; i documenti ne costituiscono i nodi. La caratteristica principale di un ipertesto è che la lettura può svolgersi in maniera non lineare.

[26] Programma che consente di usufruire dei servizi di connettività in Internet, o di una rete di computer, e di navigare sul World Wide Web.

[27] Un software capace di funzionare da solo o in maniera indipendente da altri oggetti o software, con cui potrebbe altrimenti interagire.

internet. Un terzo tipo di interfaccia grafica riguarda quelle personalizzate che lavorano su software specifici creati *ad hoc* e che non sono basate su web browser.

Ognuno dei differenti approcci scelti alla progettazione dell'interfaccia presenta dei vantaggi e degli svantaggi. Per quanto riguarda le versioni *browser based* abbiamo il vantaggio che il browser è già installato sul dispositivo utilizzato per la visualizzazione e anche un utente poco esperto ha già familiarità con questo programma e non deve fare altro che avviarlo per poter iniziare a lavorare. Affidandoci a questo tipo di visualizzazione però si ha lo svantaggio che spesso in altri sistemi operativi un determinato browser non sia presente e quindi si limiti la visualizzazione della pubblicazione. Inoltre, data la presenza di diversi browser (Opera, Safari, Firefox, Chrome, Internet Explorer) è possibile che concentrando lo sviluppo solamente su uno di questi si vada incontro a problemi di compatibilità dell'edizione digitale con alcuni di essi causando cosi problemi di funzionamento.

Dal punto di vista delle interfacce grafiche *stand alone* abbiamo come vantaggio il fatto di lavorare su un software personalizzato che offre una maggiore flessibilità nella progettazione e una maggiore libertà nell'implementare strumenti e funzioni a disposizione dell'utente. Lavorando in questa maniera però si potrebbe incorrere nel problema della compatibilità con i dispositivi al passare del tempo, ovvero si potrebbe manifestare il rischio che con l'uscita di un nuovo sistema operativo per esempio, un'interfaccia grafica risulti inutilizzabile per intero o in parte delle sue funzionalità. Dal momento che la realizzazione di un progetto digitale richiede una gran quantità di lavoro e ha dei costi di realizzazione, questo è un dettaglio da non sottovalutare per non ritrovarsi con un pubblicazione che nel giro di qualche anno risulti non accessibile ad una parte di utenza.

Quando si sviluppa una pubblicazione digitale bisogna tenere conto di alcuni criteri per rendere più facile ed intuitivo l'utilizzo di quest'ultima. Bisogna essere coerenti nell'impostazione delle pagine: ad esempio determinate azioni si devono svolgere tutte nella stessa maniera oppure determinati strumenti devono essere posizionati nel medesimo posto all'interno dell'interfaccia. Il testo della pubblicazione deve essere ben leggibile, per questo bisogna scegliere un tipo di carattere e una grandezza adeguati che si possano adattare su ogni dispositivo di visualizzazione. Bisogna tenere conto anche dei caratteri speciali, molto frequenti nei

testi antichi, e della loro visualizzazione a schermo dal momento che spesso non sono presenti tra i caratteri standard del sistema operativo in uso e quindi il progettista deve trovare una soluzione per ovviare al problema. Bisogna prestare attenzione anche al colore dello sfondo e soprattutto distinguere i diversi tipi di testo come titolo, paragrafi e citazioni tra di loro. L'utente dovrebbe avere il pieno controllo della sua finestra di lavoro e la navigazione all'interno di essa deve essere facile ed intuitiva, l'utente deve avere sempre la consapevolezza di dove si trova e con quali impostazioni e strumenti sta lavorando; questa facilità di utilizzo si può ottenere posizionando collegamenti ipertestuali, pulsanti di navigazioni e barre di scorrimento nelle medesime posizioni all'interno di tutta la pubblicazione per evitare di spostare continuamente il dispositivo di puntamento sullo schermo e creare senso di smarrimento nell'utente: per questo le funzioni principali che la pubblicazione digitale mette a disposizione dovrebbero essere sempre visibili e di facile accessibilità, magari su barre degli strumenti sempre in primo piano e le funzioni secondarie o quelle avanzate invece sarebbe buona norma inserirle all'interno di menu contestuali che le raggruppino ma che allo stesso tempo risultino di facile reperibilità da parte dell'utente.

Lo sviluppatore deve inserire all'interno dell'interfaccia una serie di strumenti che permettano all'utente di svolgere diverse operazioni durante la consultazione dell'edizione digitale, come ad esempio possibilità tramite funzione di ingrandimento di andare a zoomare il testo o la foto in questione, possibilità tramite funzioni di ricerca di ricercare determinati contenuti all'interno del testo o selezionare determinati contenuti da esaminare.

Un altro fattore importante che lo sviluppatore deve considerare è che al giorno d'oggi ci sono molti dispositivi di visualizzazione di diverse dimensioni e formato, per ciò bisognerà progettare la struttura della pubblicazioni digitale in una maniera tale che si adatti ad ogni periferica di visualizzazione. Al giorno d'oggi stanno prendendo sempre più campo supporti di visualizzazione come tablet o smartphone dove la visualizzazione di pubblicazioni digitali è ancora difficoltosa da attuare a causa delle limitate possibilità di queste periferiche. Una pubblicazione digitale deve essere ben ottimizzata in termini di richieste hardware per consentire così una migliore e più veloce visualizzazione da parte dell'utente che ne fa utilizzo.

Presa nota di tutti queste caratteristiche e requisiti si crea in principio un primo prototipo, magari anche su carta, di come dovrebbe essere strutturata la pubblicazione digitale ideando lo stile dell'interfaccia grafica per poi passare alla disposizione di strumenti, pulsanti, menu, testo ed immagini all'interno della finestra di lavoro. In un secondo momento poi si passa alla progettazione vera e propria tramite software appositi per la grafica e linguaggi di programmazione specifici per implementare tutte le funzioni che la pubblicazione dovrà mettere a disposizione. Al termine di tale lavoro si avrà una prima versione di test su cui saranno apportate tutte le migliorie e modifiche prima di arrivare alla versione definitiva fruibile dall'utente[28].

[28] Cfr. Rosselli Del Turco, Roberto. 2011. *After the editing is done: Designing a Graphic User Interface for digital editions.* Digital Medievalist 7. http://digitalmedievalist.org/journal/7/rosselliDelTurco/. (Visitato il 31 Gennaio 2015).

Capitolo 3

3. Il Vercelli Book Digitale (VBD) e EVT

3.1 Origini e obbiettivi del progetto

L'idea di creare una versione digitale del *Vercelli Book* nasce nel 2003 dal professor Roberto Rosselli del Turco, ricercatore di filologia germanica dell'Università di Torino. Ispiratosi a precedenti progetti sviluppati sul finire degli anni '90, soprattutto all'*Electronic Beowulf* di Kevin Kiernan[29], il professor Rosselli Del Turco voleva mettere a disposizione di chi doveva o voleva studiare il *Vercelli Book* un'alternativa alla consultazione diretta del manoscritto con tutte le potenzialità che le moderne edizioni digitali hanno da offrire[30].

La realizzazione di questo progetto ha richiesto una serie di strumenti adeguati per svolgere il lavoro: personal computer con discrete potenzialità di calcolo, supporti di visualizzazione adatti, scanner per la digitalizzazione delle immagini in alta risoluzione e software di fotoritocco per la manipolazione di quest'ultime, strumenti per il backup dei dati e altri software generici per lo sviluppo, il tutto gestito da uno staff di programmatori che con compiti precisi hanno dato il via al progetto con l'obbiettivo di creare un software di visualizzazione digitale chiamato EVT (*Edition Visualization Technology*) che fosse facile da utilizzare.

La realizzazione del progetto è stata suddivisa in più fasi. Una prima fase prevede la designazione degli obbiettivi che la versione digitale del *Vercelli Book* deve raggiungere: l'interfaccia grafica deve permettere all'utente la navigazione del manoscritto, spostarsi tra le pagine, visualizzarlo per intero o in parte, deve essere possibile l'ingrandimento del testo per una più dettagliata consultazione e altri strumenti che consentano specifiche operazioni sull'immagine, possibilità di selezionare ed evidenziare determinate aree visualizzando informazioni supplementari, confronto del manoscritto con una edizione diplomatica su cui sarà possibile selezionare una riga di testo ed evidenziarla nel manoscritto (e viceversa) per permettere uno studio approfondito da parte dell'utente, possibilità di confrontare il testo di un'edizione diplomatica con una critica o diverse edizioni critiche tra di

[29] Kiernan, Kevin, e Andrew Prescott. 1999. *Electronic Beowulf*. Londra, British Library.
[30] Cfr. Vercelli Book digitale. http://vbd.humnet.unipi.it/. (Visitato il 2 Febbraio 2015).

loro ponendo i due testi uno a fianco dell'altro nella medesima finestra. Una volta designati gli obbiettivi si passa alla scelta degli strumenti software da utilizzare per lo sviluppo: per garantire l'utilizzo del *Vercelli Book Digitale* su più piattaforme si è deciso di operare col linguaggio XSLT e XML affiancato dalle norme TEI (*Text Encoding Initiative*), per le immagini in alta risoluzione delle pagine del manoscritto è stato scelto il formato JPEG e per la struttura della pagina e la navigazione all'interno di essa HTML, CSS e JavaScript. Scelti i software e i linguaggi per la programmazione si è passati alla designazione dell'interfaccia, progettando dove determinati pulsanti, strumenti o menu vari dovevano essere inseriti all'interno della struttura della pagina[31].

3.2 L'interfaccia grafica di EVT

Raggiungere gli obbiettivi citati nel paragrafo precedente non è stato facile però e l'interfaccia del *Vercelli Book Digitale* è stata riprogettata diverse volte per adattarsi alle nuove esigenze che si manifestavano ma mano che si procedeva con lo sviluppo.

Per realizzare il progetto il Professor Del Turco ha studiato attentamente diversi altri progetti come l'*Electronic Beowulf* di Kiernan, lo *Junius Manuscript* o l'*Exeter Book* di B. Muir[32] arrivando a realizzare una prima versione basata su web che aveva la sola funzione di visualizzatore per le immagini e presentava diverse limitazioni come l'essere costruita con il solo utilizzo di HTML o limitata al funzionamento solo su *Internet Explorer* e non sui browser di Linux o MacOS. Nonostante fosse solo una versione preliminare si prestava comunque bene come strumento di ricerca sperimentale per gli studenti di Informatica Umanistica dell'Università di Pisa. Questa prima versione sperimentale ha dato il via allo studio di nuove versioni che, dopo svariate modifiche alle funzionalità, alla correzione di bug e miglioramenti dell'interfaccia, hanno condotto alla pubblicazione di una prima versione beta.

Questa versione è stata completamente rivista sotto l'aspetto grafico e delle

[31] Cfr. Vercelli book Digitale. http://vbd.humnet.unipi.it/?page_id=2. (Visitato il 3 Febbraio 2015).
[32] Cfr. Rosselli Del Turco, Roberto. 2014. *EVT development: an update (and quite a bit of history)*. http://visualizationtechnology.wordpress.com/.

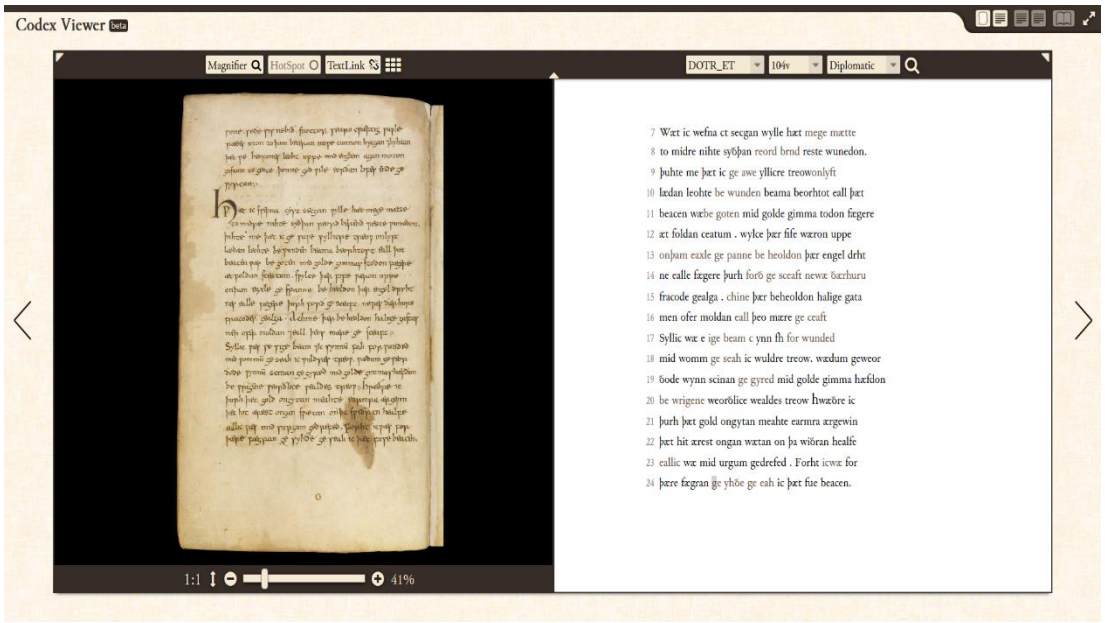
funzionalità. Al centro è presente lo spazio riservato alle immagini e al testo e presenta ai lati i pulsanti per "sfogliare" le pagine. La barra dello zoom, posta in basso nel riquadro delle immagini, è ottimizzata rispetto alle precedenti versioni: è composta da un cursore che scorre su una barra che permette di aumentare o diminuire la percentuale di zoom. E' possibile la visualizzazione a tutto schermo dei contenuti (utile per una migliore consultazione delle informazioni su supporti di piccole dimensioni). Sulla barra degli strumenti, che è possibile nascondere tramite un pulsante, sono disposti i pulsanti di tutte le funzioni che questa versione mette a disposizione.

Nel riquadro adibito alla visualizzazione delle immagini sono disponibili una serie di strumenti. Un pulsante che ci permette di visualizzare le miniature delle immagini e su cui è possibile cliccare per andare subito alla pagina desiderata. Il pulsante della funzione *hot spot* se attivato permette di evidenziare alcune aree di rilievo nelle immagini, in queste aree è poi possibile cliccare con il mouse per aprire un *pop-up*[33] che fornisce informazioni supplementari in merito all'area selezionata. Il pulsante successivo riguarda la funzione *magnifier* che se attivata apre un riquadro con un alto livello di zoom che è possibile spostare sull'immagine così da poter esaminare anche i più piccoli particolari nel testo. L'ultimo pulsante riguarda invece la funzione *text link* che è una delle funzioni più importanti aggiunte in questa versione, se attivata quest'ultima permette di andare a evidenziare nelle foto la parte di testo che abbiamo selezionato nel riquadro dedicato al testo così da capire dove sia posizionata nella pagina la frase che stiamo esaminando, questa funzionalità può essere impiegata anche nel verso opposto, ovvero selezionando una porzione dell'immagine è possibile sapere a quel riga di testo corrisponde nel riquadro ad esso dedicato. Anche nel riquadro di destra, quello adibito al testo, sono messi a disposizione strumenti di lavoro: è possibile mediante l'utilizzo di menu a tendina selezionare direttamente le pagine da esaminare senza dover necessariamente scorrerle tutte con gli strumenti di navigazione, un altro menu a tendina invece mette a disposizione la funzione più importante introdotta in questa ultima versione, ovvero la possibilità di selezionare quale edizione del manoscritto visualizzare nel riquadro.

[33] Finestre o riquadri dell'interfaccia grafica che compaiono automaticamente durante l'uso di un'applicazione in determinate situazioni.

Nella versione attuale di EVT è possibile lavorare su due distinte edizioni, una interpretativa[34] e una diplomatica[35], andando a selezionare una di queste edizioni è possibile cambiare la visualizzazione del testo sottostante. In relazione a questa funzione sono presenti in alto a destra dell'interfaccia altri tre pulsanti che cambiano la visualizzazione dei contenuti all'interno del frame[36] immagine/testo. Il primo è quello di default in cui si visualizza le immagini affiancate dal testo e con a disposizione tutte le funzioni descritte in precedenza. Il secondo pulsante fornisce invece la visualizzazione testo-testo che mediante l'utilizzo di menu a tendina posti sulla barra degli strumenti permette di selezionare le varie edizioni disponibili per poterle confrontare tra di loro. L'ultimo pulsante invece imposta la modalità di lettura estendendo a tutta pagina le immagini del manoscritto e mette a disposizione dell'utente le funzioni per lavorare sul testo.

Con il lavoro svolto fino ad oggi l'interfaccia grafica di questa prima versione beta del *Vercelli Book Digitale* risulta come nella figura 1:



[34] L'edizione interpretativa riproduce il testo in caratteri a stampa, ma 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.

[35] L'edizione diplomatica di un testo è la sua riproduzione fedele in caratteri a stampa (compresi eventuali errori, abbreviazioni, lettere diverse o usi non corrispondenti a quello moderno ecc.). Il nome deriva dalla disciplina della diplomatica, per la quale è fondamentale replicare con assoluta fedeltà diplomi e documenti giuridici.

[36] Un riquadro dove compaiono determinati contenuti di una pagina HTML.

C’è ancora molto lavoro da svolgere prima di arrivare ad una versione definitiva dell’interfaccia e per questo lo staff di programmazione è sempre a lavoro su nuove funzionalità da integrare e miglioramenti da apportare a EVT. Ad oggi sono al lavoro per integrare la funzione di ricerca che dovrebbe permette di svolgere ricerche all’interno dei testi presenti in EVT e la funzione *Digital Lightbox*, un potente strumento che darà all’utente la possibilità di lavorare sulle immagini andando a modificare parametri come luminosità, colore, contrasto, opacità per migliorare la resa delle immagini o cercare di migliorare la qualità di fogli danneggiati che altrimenti sarebbero impossibili da esaminare. Oltre a queste nuove funzionalità il team di sviluppo sta lavorando ad una edizione critica[37] da aggiungere a quelle già presenti nel *Vercelli Book Digitale*[38].

3.3 Come funziona EVT

Il funzionamento di EVT è incentrato sull’utilizzo di file XML codificati con le norme TEI, grazie a questa strategia è richiesta una minima configurazione di questi ultimi per generare nuove contenuti all’interno della pubblicazione.

La struttura di EVT è di tipo modulare con un unico foglio di stile chiamato `evt_builder.xsl`, al suo interno sono riportati i collegamenti a tutti gli altri fogli che attuano le trasformazioni da applicare al documento XML contenente i testi del *Vercelli Book*. Questa catena di trasformazioni ha lo scopo di generare un file denominato `index.html` che è la home page della pubblicazione e tutti gli altri file, sempre in formato HTML, che servono alla creazione della struttura dell’interfaccia. Per una maggiore precisione nel collocare i file si è deciso di organizzare le cartelle in maniera gerarchica dove tutti i file delle conversioni XSLT sono all’interno di una cartella chiamata `builder_pack`. Se vogliamo apportare modifiche all’EVT non bisogna fare altro che inserire all’interno della cartella `input_data` i file con le

[37] L'edizione critica ricostruisce il testo presumibilmente voluto dall'autore, confrontando tra loro tutti i codici che trasmettono quel testo e tentando di stabilire volta a volta la lezione autentica. L'edizione non si limita a essere puramente interpretativa ma tenta di correggere gli errori e di avvicinarsi all'originale.

[38] Cfr. Rosselli Del Turco Roberto, Giancarlo Buomprisco, Chiara Di Pietro, Julia Kenny, Raffaele Masotti e Jacopo Pugliese. *Edition Visualization Technology: A Simple Tool to Visualize TEI-based Digital Editions.* In “Journal of the Text Encoding Initiative”. http://jtei.revues.org/1077. (Visitato il 3 Febbraio 2015).

modifiche da effettuare e poi applicare il file di stile `evt_builder.xsl` al nostro file XML, precedentemente inserito, tramite appositi programmi di compilazione XML; se necessario all'occorrenza è possibile modificare le regole di trasformazione che vengono applicate andando a modificare il file `evt_builder-conf.xsl`. Una volta completata l'operazione di trasformazione all'interno della cartella `output_data` vengono generati i risultati delle nostre modifiche, eseguendo il file `index.html` possiamo così vedere tutta la struttura con le modifiche apportate. Il vantaggio di questa struttura gerarchica e basata sulle trasformazioni ci permette, se non siamo soddisfatti dei cambiamenti apportati, di cancellare i file nella cartella di output e di iniziare nuovamente da capo le modifiche[39].

All'interno della directory principale troviamo altre quattro cartelle: `css`, `fonts`, `images` e `js`. All'interno della cartella `css` troviamo i file per lo stile grafico dell'interfaccia che verranno applicati dopo le varie trasformazioni. Nella cartella `fonts` sono presenti i file per la visualizzazione dei caratteri all'interno dell'interfaccia, è molto importante in quanto non sempre tutti i tipi di carattere sono disponibili su un sistema operativo e molto spesso quando si lavora con testi antichi c'è la necessità di aggiungere caratteri speciali. La cartella `images` invece contiene le immagini che sono inserite all'interno dell'interfaccia grafica. L'ultima cartella contiene invece tutti i file JavaScript e jQuery utili al funzionamento dell'interfaccia, in questi file è contenuto tutto il codice che permette l'utilizzo delle varie funzionalità implementate, la gestione dell'interfaccia e i vari eventi che si attivano mentre si lavora nella finestra.

Grazie a questa struttura modulare delle directory e alla facilità di configurazione dei file di trasformazione, EVT potrebbe essere utilizzato non solo per la visualizzazione del *Vercelli Book Digitale* ma in futuro può essere utilizzato per la visualizzazione di altri manoscritti.

[39] Cfr. Rosselli Del Turco Roberto, Giancarlo Buomprisco, Chiara Di Pietro, Julia Kenny, Raffaele Masotti e Jacopo Pugliese. *Edition Visualization Technology: A Simple Tool to Visualize TEI-based Digital Editions.* In "Journal of the Text Encoding Initiative". http://jtei.revues.org/1077. (Visitato il 4 Febbraio 2015).

Capitolo 4

4. Le norme TEI e le trasformazioni XSLT

4.1 Introduzione alla TEI (Text Encoding Initiative)

La Text Encoding Initiative, conosciuta come TEI, è un consorzio che dal 1987 si impegna nello sviluppo di standard per la rappresentazione di testi in formato digitale. La TEI mediante l'utilizzo del linguaggio di markup XML (*eXtensible Markup Language*) mette a disposizione di chi ne ha la necessità linee guida (chiamate *Guidelines*) da seguire durante la creazione delle pubblicazioni digitali.

La prima versione della TEI, denominata P1, venne rilasciata nel 1990 e subito si introdussero nuove funzionalità che tra gli anni 1990 e '93 portarono alla pubblicazione della versione P2, nel 1994 venne rilasciata la versione P3 che venne considerata la prima vera e propria versione ufficiale. Queste prime versioni erano scritte in SGML[40] (*Standard Generalized Markup Language*) e con l'avvento del nuovo standard XML la versione P3 doveva essere aggiornata per supportare il nuovo linguaggio di markup. Venne rilasciato cosi nel 2002 la versione P4 con l'aggiornamento di compatibilità con XML. Dal momento però che la P4 era sempre basata sulle precedenti versioni in uso oramai dal '94, era necessario rinnovare i contenuti messi a disposizione; negli anni seguenti si inizia quindi a lavorare alla versione P5, pubblicata nel 2007, che migliorava tutto quello che in passato era stato fatto e aggiungeva nuove funzionalità come la possibilità di codificare caratteri speciali, descrizione dei manoscritti e bibliografie.

Le linee guida che la TEI offre permettono di mettere in evidenza in maniera adeguata determinate caratteristiche di un testo così da essere ben interpretate da un computer, tali regole sono inserite all'interno di una DTD[41] (*Document Type Definition*) che definisce quali elementi si possono utilizzare nel documento, la

[40] Metalinguaggio avente lo scopo di definire linguaggi da utilizzare per la stesura di testi destinati ad essere trasmessi ed archiviati con strumenti informatici, ossia per la stesura di documenti in forma leggibile da computer.

[41] Strumento utilizzato dai programmatori il cui scopo è quello di definire le componenti ammesse nella costruzione di un documento XML.

struttura di ognuno di questi elementi, gli attributi degli elementi e quale valore questi attributi possono avere. Questi elementi denominati marcatori, o più comunemente *tag*, vengono inseriti all'interno del testo così da generare una struttura che permetta di estrarre determinate informazioni da un qualsiasi tipo di testo indipendentemente dalla data o dal contenuto. È possibile marcare nomi, paragrafi, date, parti di un discorso, caratteri speciali, simboli ecc; più la marcatura sarà dettagliata e più sarà possibile estrarre informazioni nella versione digitale. Esistono due tipi di *tag*, quello di apertura (<tag>) e quello di chiusura (</tag>) all'interno dei quali sono inserite le porzioni di testo che vogliamo marcare come nell'esempio sotto riportato:

```
<p>Un tempo i <name type="person">Malavoglia
</name>erano stati numerosi come i sassi
della strada vecchia di <name
type="place">Trezza</name></p>
```

Una fondamentale caratteristica della TEI è quella di essere modulare e personalizzabile, questo permette agli sviluppatori di selezionare solo determinate funzionalità in base alle esigenze di lavoro, che si tratti di un testo antico, di un discorso trascritto o di un'edizione critica. Una delle personalizzazioni più conosciute è quella della TEI Lite, una versione che comprende un set di istruzioni base per lavorare su documenti molto semplici. È possibile inoltre estendere, con nuove aggiunte create *ad hoc*, il set di elementi e attributi per adattarli ad eventuali nuovi tipi di codifica.

Un testo digitale che rispetta le norme TEI è composto da due elementi principali, <teiHeader> e <text>, ognuno di questi può essere presente una sola volta all'interno della struttura.

Il <teiHeader> riguarda l'intestazione della pubblicazione e contiene i metadati[42] relativi al testo, «al suo interno sono riportate tutte le informazioni descrittive e dichiarative che andranno a comporre il frontespizio elettronico della

[42] Un'informazione che descrive un insieme di dati.

pubblicazione»[43]; questo campo può contenere al suo interno fino a quattro parti con specifici contenuti: le informazioni bibliografiche del testo digitalizzato sono all'interno dell'elemento `<fileDesc>`, le caratteristiche e i metodi della codifica sono contenuti nell'elemento `<encodingDesc>`, la descrizione non bibliografica del testo è contenuta nel `<profileDesc>` ed infine nell'elemento `<revisionDesc>` trovano spazio l'elenco delle modifiche apportate alla pubblicazione elettronica. Di questi campi descritti l'unico obbligatorio in ogni intestazione è il `<fileDesc>`; sono presenti anche qua campi facoltativi e tre obbligatori, questi ultimi sono: `<titleStmt>`, contenente informazioni sul titolo dell'opera e le responsabilità del contenuto intellettuale della pubblicazione (autore, editore, chi ha codificato i testi ecc); `<publicationStmt>`, raggruppa le informazioni riguardo la pubblicazione e distribuzione della pubblicazione elettronica, e l'elemento `<sourceDesc>` che fornisce descrizioni relative alle fonti da cui nasce la pubblicazione e al suo interno è presente l'elemento `<msDesc>` che dà la possibilità di inserire una gran quantità di informazioni molto dettagliate su uno o più manoscritti[44].

Tornando ad analizzare gli elementi principali, il secondo che deve essere presente all'interno di una pubblicazione è l'elemento `<text>` dove è riportato il testo vero e proprio del documento con tutti i campi utili alla codifica in digitale.

Una struttura che rispetti le norme TEI con gli elementi obbligatori è la seguente:

```
<TEI>
 <teiHeader>
  <fileDesc>
   <titleStmt>
    <title>Titolo</title>
   </titleStmt>
   <publicationStmt>
    <p>Informazioni sulla pubblicazione</p>
   </publicationStmt>
   <sourceDesc>
    <p>Informazioni sulle fonti</p>
```

[43] Cfr. TEI: Text Encoding Initiative, *TEI P5 Guidelines*. http://www.tei-c.org/release/doc/tei-p5-doc/it/html/ref-teiHeader.html. (Visitato il 20 Marzo 2015).
[44] Cfr. TEI: Text Encoding Initiative, *TEI P5 Guidelines*. http://www.tei-c.org/release/doc/tei-p5-doc/en/html/HD.html. (Visitato il 20 Marzo 2015).

```
    </sourceDesc>
   </fileDesc>
  </teiHeader>
  <text>
   <body>
    <p>Testo della pubblicazione</p>
   </body>
  </text>
 </TEI>
```

4.2 Trasformazione di un documento XML

Una volta creato il documento XML che rispetti le norme di codifica TEI, si possono elaborare i dati che contiene in modo da permetterne la visualizzazione; tale processo si attua mediante l'utilizzo del linguaggio di trasformazione XSLT (*eXtensible Stylesheet Language Transformations*). Questo tipo di file di testo usa la solita sintassi dei fogli XML ed è in grado di leggere e scrivere la struttura ad albero di questi ultimi. Un foglio di stile XSLT riporta la seguente intestazione al cui interno si sviluppa tutto il codice:

```
<?xml version="1.0" encoding="UTF-8"?>
 <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  ..
</xsl:stylesheet>
```

Il foglio contiene delle regole (*template*) dichiarative, cioè regole che specificano cosa deve essere fatto quando si incontra un determinato modello nel file di input. L'applicazione di questi *template* permette la trasformazione di un documento XML in un altro documento XML oppure in un altro documento di testo riconosciuto da un browser web; tale processo viene svolto da un processore XSLT molto spesso integrato in programmi di *editing*.

Il processo di trasformazione necessita dell'utilizzo di due file in input: il file .xml con i contenuti da trasformare e il file .xsl con le regole di trasformazione da

applicare. Il processore XSLT legge il documento XML creando un albero corrispondente e inizia a percorrere i nodi, mediante l'utilizzo della sintassi xPath[45], confrontandoli con le regole presenti nel foglio di stile. Quando si riscontra una corrispondenza (*matching*) tra un nodo del file XML e una regola scritta nel foglio di stile, il processore applica i cambiamenti descritti dalla regola e passa ad analizzare il nodo successivo fino alla fine del documento. Una volta che questo processo di controllo e trasformazione sarà completato avremo come risultato un nuovo documento XML con applicate le opportune trasformazioni.

Prendiamo per esempio queste poche righe di codice XML:

```
<cd>
 <titolo>Vita spericolata</titolo>
 <artista>Vasco Rossi</artista>
</cd>
```

e applichiamo queste regole di trasformazione XSLT:

```
<xsl:template match="/">
 <html>
  <body>
   <div class=”table”>
    <div class=”row”>
     <div class=”title”>Titolo</div>
     <div class=”title”>Artista</div>
    </div>
   <xsl:for-each select="cd">
    <div class=”row”>
     <div class=”left”><xsl:value-of select="titolo"/></div>
     <div class=”right”><xsl:value-of select="artista"/></div>
    </div>
   </xsl:for-each>
   </div>
```

[45] Linguaggio che permette, mediante una sintassi specifica, l'individuazione dei nodi all'interno di un documento XML.

```
  </body>
 </html>
</xsl:template>
```

Il processore XSLT procederà nella trasformazione e darà come risultato una pagina HTML con una tabella composta da tre righe e due colonne: nella colonna di sinistra è riportato il titolo e in quella di destra l'artista come in figura 1:

Titolo	Artista
Gli anni	Max Pezzali
Vita spericolata	Vasco Rossi

Figura 1: il risultato del foglio di stile XSLT applicato al codice XML.

Capitolo 5

5. Manipolazione delle informazioni del VBD e visualizzazione in EVT

5.1 Introduzione alla progettazione

Con l'ausilio delle procedure e degli strumenti hardware e software descritti nei capitoli precedenti gli sviluppatori hanno creato il file XML, contenente tutta la struttura del *Vercelli Book*, e l'interfaccia grafica che ospiterà i contenuti della pubblicazione. All'interno del file XML sono stati inseriti tutti gli elementi necessari per la codifica di questo tipo di manoscritti così da avere a disposizione un ampio set di informazioni da gestire. La marcatura di tutte le informazioni indispensabili alla conversione digitale è stata effettuata secondo le regole standard dei moduli TEI e senza particolari personalizzazioni per rendere il più generico possibile il codice.

In questo capitolo verrà trattato nello specifico lo sviluppo del codice per la visualizzazione e l'integrazione dei moduli `<teiHeader>` e `<msDesc>` all'interno di EVT, verranno processati tutti i loro elementi interni che costituiranno il frontespizio elettronico e riporteranno tutte le informazioni descrittive della pubblicazione digitale. Questi set di informazioni sono molto importanti all'interno di una pubblicazione perché permettono all'utente di documentarsi sull'origine del testo, sulla codifica, sui responsabili del lavoro, sulle informazioni bibliografiche e sulle eventuali modifiche apportate. All'interno del modulo `<msDesc>` sono inserite tutte le informazioni dettagliate sul manoscritto che costituiscono la parte fondamentale della pubblicazione su cui l'utente potrà basare i suoi studi e ricerche[46].

Come già noto dai paragrafi precedenti EVT offrirà la possibilità di visualizzare più di un manoscritto, per questo motivo in accordo con il Professore è stato deciso di separare le informazioni generiche sulla pubblicazione, contenute nel `<teiHeader>`, da quelle specifiche del manoscritto contenute in `<msDesc>`. Dato l'utilizzo della versione 1.0 di XSLT, che non consente output multipli da un solo

[46] Cfr. TEI P5 Guidelines. http://www.tei-c.org/release/doc/tei-p5-doc/en/html/MS.html. (Visitato il 21 Marzo 2015).

foglio, ho sviluppato il codice in due fogli di stile diversi, uno per trasformare i contenuti di `<teiHeader>` e l'altro per quelli di `<msDesc>`.

5.2 Gestione dei dati nei moduli <teiHeader> e <msDesc>

Nonostante abbia sviluppato due fogli di stile separati, ho lavorato su questi con il medesimo approccio per cui il codice che andrò a commentare avrà le medesime caratteristiche e metodologie di sviluppo in entrambi i fogli; per questa ragione negli esempi al posto del nome specifico degli elementi ne userò uno generico.

Come primo passo ho prodotto un *template* che costituirà il corpo della pagina HTML in cui verranno visualizzate tutte le informazioni:

```
<xsl:template name="NOME_TEMPLATE">
 <html>
  <body>
  <div id=”NOME_ID”>
   <xsl:apply-templates select="//tei:ELEM"/>
  </div>
 </body>
</html>
</xsl:template>
```

Il comando `<xsl:template>` contiene le regole che verranno eseguite dal comando `<xsl:apply-templates>`; nel nostro caso `<xsl:template name="NOME_TEMPLATE">` contiene le regole per generare la struttura della pagina HTML con all'interno le informazioni che il comando `<xsl:apply-templates select="//tei:ELEM"/>` ottiene quando il processore XSLT trova all'interno del file XML il nodo specificato dall'attributo `select`.

Una volta recuperate ed inserite nel corpo della pagina, le informazioni vengono rappresentate in un unico blocco di testo senza alcuna formattazione, per cui

è necessario dare ordine a questi contenuti suddividendoli in maniera adeguata. Questa organizzazione viene strutturata secondo le regole dei quattro elementi fondamentali di cui è composto il `<teiHeader>`: il `<fileDesc>`, l'`<encodingDesc>`, `<profileDesc>` e `<revisionDesc>`. Per gestire una prima suddivisione in blocchi delle informazioni ho creato il seguente *template*:

```
<xsl:template match="tei:ELEM">
 <div id=”NOME_ID”>
  <xsl:apply-templates select="tei:ELEM"/>
  <xsl:apply-templates select="tei:ELEM"/>
  ..
  ..
 </div>
</xsl:template>
```

Una volta che il processore XSLT trova l'elemento specificato dall'attributo `match` del comando `<xsl:template>`, genera le regole da applicare. In questo caso verrà creato all'interno della pagina HTML un `<div>`[47] (box), con lo stesso nome dell'elemento, in cui verranno visualizzate le informazioni che il comando `<xsl:apply-templates>` trasformerà una volta trovati gli elementi specificati dall'attributo `select`. A questo punto avremo una prima suddivisione in blocchi della pagina HTML differenziando il contenuto in base al tipo di informazione.

Questi quattro elementi principali a loro volta possono contenere altre sottocategorie di elementi al loro interno. Come spiegato nel capitolo precedente ne è un esempio l'elemento `<fileDesc>` che contiene altri elementi che delineano determinati tipi di informazione; tali elementi vengono organizzati nella medesima maniera sempre all'interno di un `<div>` identificabile da un `ID` che riporta il nome dell'elemento in questione. Pertanto il *template* per la trasformazione sarà identico a quello precedente con la sola differenza che andremo a trasformare elementi con nome diverso. Questo tipo di approccio genera un insieme di "scatole" annidate che

[47] Sezione di una pagina HTML, più precisamente un contenitore.

rispecchiano la struttura ad albero del documento XML, in questo modo sarà possibile gestire separatamente ogni set di informazioni nella maniera più appropriata.

Una volta definita la struttura dei contenuti sono passato poi alla gestione delle informazioni vere e proprie eseguita dal seguente codice:

```
<xsl:template match="tei:ELEM">
 <div id="ID_NAME">
  <h3>TITOLO_SEZIONE</h3>
  <div class=table”>
   <xsl:if test="tei:ELEM and tei:ELEM/normalize-space()">
    <div class=row”>
     <div class="left_col">
      TITOLO_RIGA:
     </div>
     <div class=”right_col”>
      <xsl:value-of select="tei:ELEM"/>
     </div>
   </xsl:if>
   ..
   ..
  </div>
 </div>
</xsl:template>
```

Come in precedenza il comando `<xsl:template match="tei:ELEM">` viene utilizzato per individuare i nodi su cui vanno applicate le regole del *template*. Una volta individuato il nodo, si passa all'esecuzione di `<xsl:value-of select="tei:ELEM"/>`. Il comando `<xsl:value-of>` ha la funzione di restituire tutti i caratteri che si trovano tra i *tag* di apertura e chiusura dell'elemento selezionato dall'attributo `select`, in questo caso quindi si avrà come risultato l'informazione vera e propria da visualizzare.

Dal momento che il file XML del *Vercelli Book* riporta tutti gli elementi standard per un documento di questo genere, si può verificare il caso in cui uno o più di questi campi non contengano informazioni e quindi il foglio di stile XSLT processi e visualizzi nella pagina HTML finale questi campi come campi vuoti. Per evitare questo inconveniente all'interno di ogni *template* ho inserito un controllo mediante la funzione `<xsl:if>`:

```
<xsl:if test="tei:ELEM and tei:ELEM/normalize-space()">
```

Il comando `<xsl:if>` ha lo scopo di controllare se si verifica o no una certa condizione, in questo caso si accerta che si verifichino due condizioni congiunte: `test="tei:ELEM"` effettua un controllo sul file XML e verifica che sia presente l'elemento a cui dobbiamo applicare la regola, `tei:ELEM/normalize-space()` invece controlla che l'elemento non sia vuoto. Queste due condizioni unite dall'operatore logico `and` fanno sì che non siano processati elementi mancanti o vuoti così da non comparire nell'output della pagina HTML finale.

Una volta inserite nell'output le informazioni però non risultano ancora ben strutturate e rendono difficoltosa l'interpretazione, per questo motivo ho deciso di inserirle all'interno di tabelle composte da due colonne: nella colonna di sinistra si riporta il nome dell'informazione e in quella di destra la descrizione:

```
<div class=TABELLA”>
 <div class=”RIGA”>
  <div class=”COLONNA SX”>
    NOME_INFORMAZIONE
  </div>
  <div class=”COLONNA DX”>
    <xsl:value-of select=”tei:ELEM”/>
  </div>
 </div>
</div>
```

Per delineare questa struttura ho inserito tra le righe di codice XSLT *tag* HTML che grazie alle funzionalità della versione 5 permette di gestire le tabelle mediante l'utilizzo del *tag* `<div>` a cui è possibile attribuire classi per definire tabelle, righe e colonne; in questa maniera ho potuto inserire il comando `<xsl:value-of select="tei:ELEM"/>` nella colonna di destra della tabella in modo da permettere una migliore lettura dell'informazione. Lo stile di visualizzazione, la struttura della tabella ed eventuali altri dettagli saranno poi gestiti nel foglio di stile CSS.

Questa sopra riportata è la struttura generale del foglio XSLT che trasforma la maggior parte dell'XML in ingresso. Sono presenti però all'interno della struttura XML più elementi con lo stesso nome; in questo caso il comando `<xsl:value-of>` restituisce solo il risultato del primo elemento che incontra per cui ho dovuto necessariamente utilizzare il comando `<xsl:for-each>` che ha la funzione di processare in maniera ricorsiva più elementi con lo stesso nome:

```
<xsl:for-each select="tei:ELEM">
 <xsl:value-of select="tei:ELEM"/>
</xsl:for-each>
```

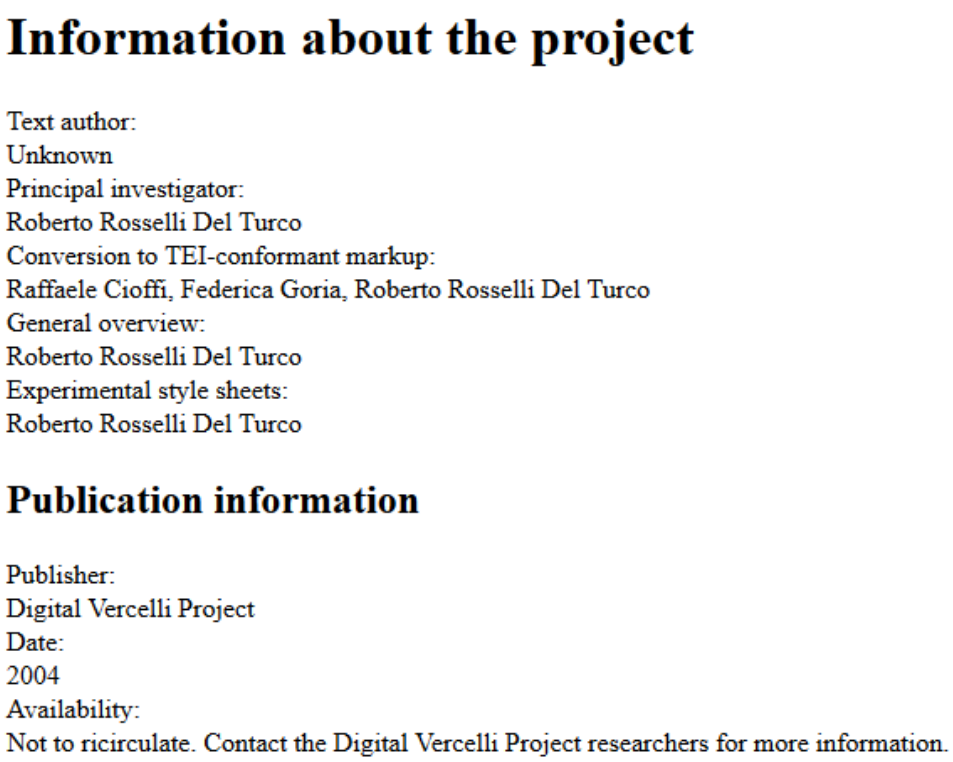
All'interno del foglio di stile XSLT sono presenti poi altri template che definiscono stili di visualizzazione particolari o necessari per questo tipo di pubblicazione come il seguente:

```
<xsl:template match="tei:title">
 <xsl:choose>
  <xsl:when test="./@level='m'">
   <span class="italic">
    <xsl:apply-templates/>
   </span>
  </xsl:when>
 </xsl:choose>
```

```
</xsl:template>
```

L'elemento `<xsl:choose>` viene utilizzato insieme a `<xsl:when>` per verificare una o più condizioni. In questo caso `<xsl:when>` verifica che l'attributo `level` dell'elemento `title` abbia come valore "`m`", se la condizione è soddisfatta verrà applicata una classe "`italic`" (solitamente utilizzata per definire il testo in corsivo) che in seguito sarà gestita dal CSS come testo corsivo.

Giunto a conclusione della progettazione del codice XSLT l'ho applicato sul file XML del *Vercelli Book Digitale* che mi ha generato la pagina HTML come nella figura 1:



Information about the project

Text author:
Unknown
Principal investigator:
Roberto Rosselli Del Turco
Conversion to TEI-conformant markup:
Raffaele Cioffi, Federica Goria, Roberto Rosselli Del Turco
General overview:
Roberto Rosselli Del Turco
Experimental style sheets:
Roberto Rosselli Del Turco

Publication information

Publisher:
Digital Vercelli Project
Date:
2004
Availability:
Not to ricirculate. Contact the Digital Vercelli Project researchers for more information.

Figura 2: Per questioni di spazio è stata riportata solo una parte delle informazioni.

5.3 Integrazione all'interno dell'interfaccia grafica

Una volta che le informazioni sono state processate e organizzate per la visualizzazione, il passo successivo è stato quello di integrarle nell'interfaccia di EVT e dal momento che si trattano due tipi diversi di informazioni ho deciso (discutendone con il professore e il team di sviluppatori) di visualizzarle in due modi e posti differenti. Per quanto riguarda le informazioni del manoscritto la decisione è stata quella di farle comparire nel riquadro delle immagini; premendo sul pulsante *MS info* (posizionato a fianco dei pulsanti *Magnifier*, *Hot-Spot* e *Text-Link*) è possibile aprire o chiudere il riquadro delle informazioni in stile menu a tendina. Alle informazioni generiche si accede invece tramite il pulsante *[i]* posto in alto a destra vicino ai pulsanti adibiti al cambio della vista; anche qua si apre un riquadro contenente tutte le informazioni.

L'implementazione di queste soluzioni ha richiesto la modifica di diversi fogli XSLT che gestiscono le trasformazioni atte a generare la struttura HTML di EVT. Il primo passo è stato quello di includere all'interno del file `evt_builder.xsl`, che contiene i riferimenti di tutti i fogli di trasformazione del software EVT, il comando per il riferimento ai miei due fogli di stile XSLT, chiamati `evt-MS_Info_Visualization.xsl` e `evt-Header_Info_Visualization.xsl`:

```
<xsl:include href="modules/fundamental_units/evt-
MS_Info_Visualization.xsl"/>
<xsl:include href="modules/fundamental_units/evt-
Header_Info_Visualization.xsl"/>
```

Il secondo passo ha richiesto la modifica del file `evt_builder-conf.xsl`, al cui interno sono presenti tutte le variabili necessarie per la personalizzazione dell'output HTML, con la dichiarazione di due variabili booleane[48],

[48] Una variabile che assume valori del tipo "Vero/Falso", "Ture/False" o "1/0".

`msDescription` e `HeaderInformation`, entrambe settate con un valore `true` e indispensabili per la trasformazione degli elementi `<teiHeader>` e `<msDesc>`:

```
<xsl:param name="msDescription" select="true()"/>
<xsl:param name="HeaderInformation" select="true()"/>
```

Queste due variabili sono quindi strettamente connesse al codice inserito all'interno del file `evt_builder-main.xsl`, che è il medesimo per entrambe le variabili, con lo scopo di generare i file HTML con il risultato delle trasformazioni:

```
<xsl:if test="$NOME_VARIABILE=true()">
 <xsl:result-document method="html" encoding="UTF-8"
  media-type="text/plain" byte-order-mark="yes"
  href="{$filePrefix}/data/output_data/information/FILE.html"
  <xsl:call-template name="NOME_TEMPLATE">
  </xsl:call template>
 </xsl:result-document>
</xsl:if>
```

Nella prima riga del codice si esegue subito il controllo della variabile impostata in precedenza e dal momento che è stata settata con il valore `true` il codice passa alla creazione del file HTML e lo posizionerà nella cartella `information` all'interno della directory di EVT; in seguito poi viene richiamato il *template* che andrà a contenere tutte le regole per la trasformazione delle informazioni.

Per una corretta integrazione ho dovuto aggiungere al file `evt_builder-structure.xsl` le seguenti righe di codice:

```
<xsl:if test="$msDescription=true()">
 <MS_Info active="1"/>
</xsl:if>
<xsl:if test="$HeaderInformation=true()">
```

```
 <Header_Info active="1"/>
</xsl:if>
```

Questa semplice condizione se soddisfatta andrà ad inserire all'interno del file `structure.xml` le righe `<Header_Info active="1"/>` e `<MS_Info active="1"/>` utili al codice JavaScript per gestire gli eventi di visualizzazione del contenuto all'interno dell'interfaccia.

A questo punto l'ultimo file XSL da modificare è il file `evt_builder.callhtml.xsl`. All'interno di questo file sono presenti tutte le regole per la creazione della struttura HTML a cui ho aggiunto il codice per generare il box che andrà a contenere le informazioni generiche della pubblicazione e quelle del manoscritto:

```
<xsl:if test="$NOME_VARIABILE=true()">
 <div id="NOME_ID"></div>
</xsl:if>
```

Inoltre, nel medesimo file, viene anche inserito il codice per la creazione dei due pulsanti cliccabili per aprire e chiudere i pannelli delle informazioni:

```
<xsl:if test="$NOME_VARIABILE=true()">
 <span class="mainButtons" id="NOME_ID"
 title="TITOLO_PULSANTE">
  <span>NOME_PULSANTE</span>
 </span>
</xsl:if>
```

Per il funzionamento e posizionamento dei pulsanti e box all'interno della pagina, questi due set di istruzioni devono essere posizionati correttamente tra le righe di codice del file che genera la struttura cosi da farli comparire nelle sezioni più appropriate della pagina finale.

Tutto il funzionamento dei due pulsanti e degli eventi attribuiti ad essi è gestito dal codice JavaScript contenuto all'interno del file `interface_control.js` (riporto solo il codice inerente al funzionamento di un solo pulsante dal momento che il codice è il medesimo per entrambi):

```
if (($(xml).find('MS_Info').length > 0) &&
   ($(xml).find('MS_Info').attr('active')==1)){
$('#MS_Info').load("data/output_data/information/MS_Info.html
#MS_Info_cont");
$('#switchMS_Info').click(function(){
 if($('#MS_Info').is(':visible')){
    $('#MS_Info').hide('drop', {direction:'up'}, 'linear');
    $('#switchMS_Info').removeClass('active');
 } else {
    $('#Ms_Info').show('drop', {direction:'up'}, 'linear');
    $('#switchMs_Info').addClass('active');
 }
 });
```

La prima riga di codice esegue un doppio controllo per verificare che sia presente nel file `structure.xsl` la riga di codice `<MS_Info active="1"/>` e che l'attributo `active` sia impostato a "1", a questo punto se la condizione viene soddisfatta viene caricato il contenuto del file `MS_Info.html` all'interno del `<div>` MS_Info tramite il seguente comando (lo stesso viene fatto per il file `Header_Info.html` all'interno del `<div>` Header_Info):

```
$('#MS_Info').load("data/output_data/information/MS_Info.html
#MS_Info_cont");
```

Successivamente si passa alla gestione dell'evento "click" sul pulsante:

```
$('#switchMS_Info').click(function(){
```

```
if($('#MS_Info').is(':visible')){
   $('#MS_Info').hide('drop', {direction:'up'}, 'linear');
   $('#switchMS_Info').removeClass('active');
} else {
   $('#Ms_Info').show('drop', {direction:'up'}, 'linear');
   $('#switchMs_Info').addClass('active');
}
```

La condizione ci permette di verificare se i pannelli delle informazioni sono aperti o chiusi: se sono visualizzate le informazioni al momento del click queste verranno nascoste con un'animazione in stile menu a tendina e verrà eliminata la classe `active` al pulsante, viceversa, se le informazioni non sono visualizzate, cliccando sul pulsante verrà aperto il pannello e verrà attribuita al pulsante la classe `active`.

A questo punto della progettazione l'ultimo passaggio è stato gestire lo stile grafico delle informazioni tramite CSS. Avendo lavorato su due differenti fogli XSLT ho preferito mantenere separato anche il codice CSS delle informazioni generiche da quelle specifiche dei manoscritti con i file chiamati rispettivamente `Header_Info.css` e `MS_Info.css`, ho preferito questa scelta per evitare di appesantire ulteriormente il file CSS generale e per rendere più facili e veloci eventuali modifiche al codice. I due fogli in questione presentano solo alcune personalizzazioni e per il resto si rifanno al CSS generale della pubblicazione. Per prima cosa ho definito le dimensioni e la posizione del box delle informazioni impostando anche la distanza dei contenuti dai margini e ho reso nascoste di default le informazioni; il colore delle sfondo, la dimensione e il tipo di carattere riprendono quelli di tutta la pubblicazione per mantenere una coerenza di stile. Particolare attenzione l'ho prestata alla disposizione e visualizzazione delle informazioni all'interno delle tabelle. Ho definito grandezza e posizione della tabella e ho gestito i margini e spaziature delle righe e colonne in maniera da avere una visualizzazione chiara delle informazioni, i titoli delle voci sono visualizzati con il carattere "grassetto" per una più veloce interpretazione e sono separati dalle informazioni con una linea verticale per una maggiore distinzione. Dal momento che queste informazioni sono state suddivise in blocchi, ad ogni blocco è stato attribuito un

titolo per distinguere i vari tipi di informazione tra di loro. Siccome sono presenti una buona quantità di informazioni da visualizzare, e non è possibile visualizzarle tutte all'interno degli spazi, si è reso opportuno inserire una barra di scorrimento verticale per permettere la totale visualizzazione.

Inserito nell'apposita cartella nella directory di EVT, il foglio è richiamato con il seguente collegamento all'interno del file `evt_builder.callhtml.xsl`:

```
<link rel="stylesheet" type="text/css"
href="{$html_path}/css/NOME_FILE.css" />
```

Una volta applicato il foglio di stile CSS, il risultato visivo è quello della fig. 3:

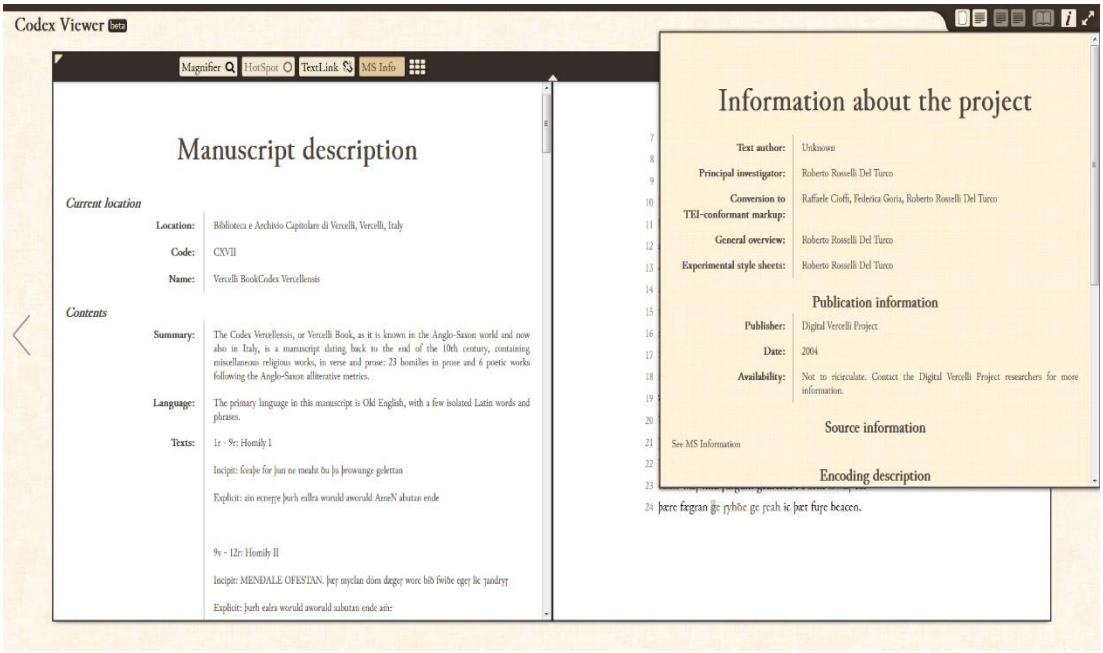


Figura 3: Finestra EVT con i pannelli di informazione aperti.

Conclusioni

Pur non essendo ancora una versione definitiva, questa beta di EVT mette a disposizione degli utenti un buon set di strumenti per studiare i testi del *Vercelli Book* e a tal proposito può essere considerata una solida base di partenza per ricercatori e sviluppatori che volessero integrare nuove funzionalità e nuovi strumenti. Da questo punto di vista il team di sviluppo del Professor Rosselli Del Turco è sempre al lavoro per apportare modifiche e migliorie al codice e sono in corso di sviluppo una serie di nuovi strumenti, come la funzione di ricerca o la Digital Lightbox, da integrare nell'interfaccia in un futuro prossimo.

Come spiegato nel capitolo tre il punto di forza di EVT è la sua modularità che offre la possibilità di mantenere una impostazione standard che consente le modifiche al progetto mediante l'aggiunta, o l'eliminazione, di file di contenuti personalizzati senza la necessità di dover riscrivere tutto il codice ogni volta che si effettuano cambiamenti. È questa caratteristica che mi ha permesso di lavorare al progetto sviluppando il mio codice su file separati da quelli già presenti senza incorrere nel rischio di causare malfunzionamenti a contenuti già esistenti; in questa maniera quando c'è stata la necessità di apportare correzioni o profonde modifiche al codice a causa di errori di progettazione, non ho fatto altro che rimuovere i miei file e ricominciare da capo aggiungendo altri file con un nuovo codice scritto.

Grazie alle informazioni del modulo `<teiHeader>` fornitemi dal Professor Rosselli Del Turco ho potuto lavorare su una buona quantità di informazioni che, come spiegato in dettaglio nel capitolo cinque, ho inserito all'interno di EVT integrando nel migliore dei modi il mio lavoro all'interno dell'interfaccia in modo tale da riprendere lo stile e le meccaniche di tutta la pubblicazione.

Questa scelta di impostazioni offrirà in futuro la possibilità di visualizzare le informazioni relative a più manoscritti in quanto lo scopo principale di EVT è quello di essere un visualizzatore di pubblicazioni digitali e non solo esclusivamente un visualizzatore per il *Vercelli Book*.

Bibliografia

Capochiani, Francesca. 2015. Vercelli Book Digitale. http://vbd.humnet.unipi.it/.

Ciotti, Fabio (a cura di). 2004. *TEI Lite: Introduzione alla codifica dei testi*. http://www.tei-c.org/Vault/P4/Lite/teiu5_it.html.

Encyclopædia Britannica, voce Cynewulf. http://www.britannica.com/EBchecked/topic/148420/Cynewulf.

Halsall, Maureen. 1969. *Vercelli and the "Vercelli Book"*. In "PMLA", Vol. 84, No. 6 (Ott. 1969), pp. 1545-1550. (http://www.jstor.org/stable/1261500).

Introvigne, Massimo. 2010. *Vercelli Book. La mostra e l'intervento di Introvigne su "L'Osservatore Romano"*. http://www.cesnur.org/2010/mi-ver.html.

Kiernan, Kevin, e Andrew Prescott. 1999. *Electronic Beowulf*. Londra, British Library.

Luiselli Fadda, Anna Maria, 1999. *Tradizioni manoscritte e critica del testo nel Medioevo germanico*. Bari, Laterza.

Microsoft Developer Network. *XSLT Reference*. 2015. https://msdn.microsoft.com/en-us/library/ms256069%28v=vs.110%29.aspx.

Muir, Bernard James, e Bodleian Library. 2004. *A digital facsimile of Oxford, Bodleian Library, MS. Junius 11*. Oxford: Bodleian Library. http://www.digitalmedievalist.org/journal/2.1/mcgillivray/.

Roberto Rosselli Del Turco, Giancarlo Buomprisco, Chiara Di Pietro, Julia Kenny, Raffaele Masotti, and Jacopo Pugliese. *Edition Visualization Technology: A Simple*

Tool to Visualize TEI-based Digital Editions. In "Journal of the Text Encoding Initiative". http://jtei.revues.org/1077.

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

Rosselli Del Turco, Roberto. 2014. *EVT development: an update (and quite a bit of history)*. http://visualizationtechnology.wordpress.com/.

Text Encoding Initiative, TEI P5 Guidelines. http://www.tei-c.org/Guidelines/P5/.

Vercelli Book Digitale. http://vbd.humnet.unipi.it/.

W3 School. *Tutorial*. http://www.w3schools.com/.

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