UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Bellini Digital Correspondence: ottimizzazione della fruizione web mediante EVT ed utilizzo delle tecnologie XML-based Saxon-JS2, XQuery ed eXist-db

Candidata: *Santa Pellino*
Relatore: *Prof. Angelo Mario del Grosso*
Correlatori: *Prof.ssa Marina Riccucci,*
Dott.ssa Daria Spampinato

Anno Accademico 2020-2021

Abstract

Il lavoro di tesi si concentra sulla rappresentazione dell'edizione scientifica digitale *Bellini Digital Correspondence* e sulla sua presentazione attraverso l'utilizzo del software EVT e, in aggiunta a questo, attraverso altre tecnologie innovative XML-based. L'edizione oggetto di studio contiene dati di interesse storico e culturale poiché comprende le carte digitalizzate e la trascrizione eseguita seguendo l'edizione critica di Graziella Seminara del *corpus* di lettere autografe del compositore Vincenzo Bellini conservate all'interno del Museo Civico Belliniano di Catania.

La rappresentazione del *corpus* in formato digitale avviene tramite un processo di codifica XML che segue le linee guida TEI.

L'attività ha implementato un processo utile ad ottimizzare l'estrazione di informazioni dal repertorio codificato in modo da poterlo presentare attraverso una vista web rendendolo fruibile agli utenti.

Il lavoro eseguito per la fruizione dell'edizione mediante il software di visualizzazione EVT si è suddiviso nei seguenti passaggi:

- analisi, revisione ed armonizzazione della codifica;
- analisi e studio della presentazione dell'edizione sul software;
- implementazione delle regole di trasformazione XSLT da inserire nel foglio di stile per manipolare e trasformare i documenti XML del repertorio di codifica al fine di mostrare le informazioni di interesse e di ottimizzarne la fruizione;
- utilizzo del processore XSLT SAXON che permette l'elaborazione delle istruzioni indicate nelle regole di trasformazione del foglio di stile;
- impiego del meccanismo XInclude per montare in un unico file i dati ricavati dalle trasformazioni XSLT di tutte le lettere del *corpus.*

Le regole di trasformazione hanno permesso la ristrutturazione di alcuni frammenti XML del repertorio di codifica, tra questi la registrazione delle coordinate per l'allineamento testo-immagine, la creazione dei blocchi di testo visualizzati nei pop-up associati agli hotspot, la visualizzazione della bibliografia, delle note, dei dati notevoli, delle liste delle entità citate, delle informazioni riguardanti le opere e i termini citati nelle lettere e la disposizione dei metadati codicologici.

Il lavoro condotto soddisfa gli obiettivi relativi al miglioramento della presentazione e fruizione dell'edizione digitale. Tuttavia, ha richiesto l'esecuzione di molte operazioni

e l'implementazione di complesse regole XSLT al fine di estrarre i dati utili alla fruizione dell'edizione.
In più, restano da eseguire alcuni lavori di affinamento sia per quanto riguarda la resa grafica di alcuni dati attraverso la modifica del foglio di stile CSS (ad esempio gli spazi bianchi e l'andata a capo per alcune righe) sia per quanto riguarda la presentazione di alcuni dati dell'edizione (come, per esempio, le zone rappresentate attraverso polilinee ed alcuni dati della bibliografia) implementando nuovi template XSLT da aggiungere al foglio di stile XSL.
Al fine di sperimentare ulteriori tecnologie dell'ecosistema XML sono state verificate valide soluzioni per ottenere la presentazione dei dati codificati dell'edizione digitale e sono state poste le seguenti domande a cui si è cercato di dare una risposta convincente. Esistono altre tecnologie, in aggiunta ad EVT, che ci consentono la fruizione dell'edizione digitale? Sperimentando lo sviluppo di nuove applicazioni XML-based, tra cui Saxon-JS, XQuery ed eXist-db, è possibile ottenere dei buoni risultati per la presentazione dei dati digitali?

Introduzione

Il lavoro di tesi nasce dalle esigenze emerse in seguito allo svolgimento del progetto di tirocinio curriculare nell'ambito del progetto *BellinInRete.*
L'attività è stata incentrata principalmente sullo sviluppo e sulla fruizione dell'edizione scientifica digitale delle lettere autografe belliniane *Bellini Digital Correspondence* (d'ora in poi BDC), il cui obiettivo è quello di pubblicare, seguendo pratiche rigorose, il *corpus* delle lettere manoscritte del compositore Vincenzo Bellini conservate presso il Museo Civico Belliniano di Catania.
In questo contesto il *corpus* è sviluppato mediante un processo di codifica che segue il vocabolario messo a punto dalla Text Encoding Initiative (TEI) mentre la fruizione dell'edizione si presenta via web attraverso l'utilizzo di EVT (Edition Visualization Technology): un software open-source sviluppato in contesti universitari coordinato dal prof. Roberto Rosselli Del Turco.
Lo sviluppo dell'edizione ha previsto diverse fasi redazionali a cui è seguito un ulteriore lavoro di adattamento al fine di ottimizzarne l'esperienza nell'ambiente web. Nello specifico, l'operato si attua con la personalizzazione del software di visualizzazione EVT e con la ristrutturazione di alcuni frammenti XML al fine di

rendere compatibile la fonte codificata con l'applicazione. La realizzazione di queste operazioni si ottiene attraverso un processo di trasformazione della struttura della codifica presieduta da regole XSLT definite in specifici documenti detti "fogli di stile". Tutte queste attività hanno migliorato la fruizione web dell'edizione adattando e rendendo compatibile il *corpus* al software di visualizzazione e, contestualmente, ottimizzando l'esperienza utente nel web.

Il lavoro di tesi si è sviluppato in tal modo come attività di ricerca finalizzata all'ottimizzazione della fruizione dell'edizione digitale sul web mediante lo studio delle varie regole XSLT del foglio di stile da applicare. Tale soluzione è risultata molto produttiva ed efficiente per la fruizione della risorsa.

Il lavoro di tesi si è arricchito di ulteriori sperimentazioni finalizzate allo sviluppo di prototipi per la fruizione dell'edizione del carteggio belliniano. Tali prototipi hanno fatto uso di ulteriori tecnologie XML per la pubblicazione web del *corpus*.

1. Premessa

Il Museo Civico Belliniano di Catania è ubicato all'interno del palazzo Gravina Cruylas, luogo in cui Vincenzo Bellini[1] nacque e in cui trascorse i primi sedici anni di vita. All'interno del Museo sono conservate risorse riconducibili ai settori archivistico (testuale e musicale), bibliografico e museale: per 'risorse' si intendono partiture autografe, libretti, oggetti (quali quadri, pianoforti, busti, oggetti personali del musicista), riproduzioni sonore e audiovisive, dischi, fotografie e lettere manoscritte.

Le lettere archiviate nel Museo comprendono sia quelle del carteggio belliniano, sia quelle del Fondo Perucchini.[2]

L'epistolario belliniano è catalogato con la segnatura *LL* e consta di 234 lettere tra cui:

- 40 lettere autografe di Vincenzo Bellini catalogate con la segnatura *LL1;*
- 85 lettere destinate a Bellini catalogate con la segnatura *LL2*;

[1] Vincenzo Bellini (Catania, 3 novembre 1801 – Puteaux, 23 settembre 1835) è stato un compositore italiano, tra i più celebri operisti dell'Ottocento.

[2] Il Fondo Perucchini comprende un vasto carteggio accumulato dal compositore e pianista Giovanni Battista Perucchini (1784-1869), famoso nelle accademie filarmoniche veneziane. Il fondo fu acquistato nel 1998 dal Comune di Catania vista la presenza di due lettere intercorse con Vincenzo Bellini.

- 45 lettere scritte da Rosario Bellini[3] dopo la morte di Vincenzo catalogate con la segnatura *LL3*;
- 50 lettere inviate alla famiglia di Bellini dopo la sua morte catalogate con la segnatura *LL4*;
- 14 lettere di altri mittenti e destinatari catalogate con la segnatura *LL5*.

Il Carteggio Perucchini è catalogato invece con la segnatura *CP* ed è costituito da 709 lettere e comprende due lettere scritte da Bellini (già conteggiate nell'elenco *LL*), di cui una scritta a Milano il 25 novembre 1830 e l'altra scritta a Napoli il 28 gennaio 1832.[4] Attualmente, al Museo Civico Belliniano di Catania, sono conservate 40 lettere autografe di Bellini che fanno parte del carteggio belliniano. Queste sono state scritte in un periodo di tempo che va dal 3 maggio 1819, con la *supplica* inviata a Stefano Notarbartolo, al 3 settembre 1835, con la bozza della lettera indirizzata a Giovanni Ricordi. Molte di esse provengono da collezioni private, altre, che fanno riferimento al suo soggiorno a Parigi negli ultimi anni della sua vita, invece, furono trasmesse da Rossini alla sua famiglia dopo la morte del musicista.

Le prime lettere di questa raccolta furono acquisite dalla donazione effettuata nel 1930 dal maestro Ascanio Bazan, pronipote del compositore, mentre le altre furono recepite in seguito. Tuttavia, vista l'assenza di un inventario dei beni archivistici di proprietà del Museo Civico Belliniano, è impossibile determinare con esattezza la data di acquisizione di ciascuno di questi documenti. Possiamo però risalirvi attraverso la consultazione delle informazioni contenute in alcuni cataloghi storici dello stesso periodo. Nel 1935, anno di pubblicazione del secondo catalogo redatto da Benedetto Condorelli,[5] la collezione comprende 25 documenti. Segue poi un'acquisizione di altre sette lettere in un inventario redatto nel 1968. Nel 1998 si registra poi un'altra acquisizione di un piccolo gruppo di lettere quando il Comune di Catania acquista una raccolta epistolare messa in vendita da Christie's, appartenente a Giovanni Battista Perucchini, catalogata ora nel "Fondo Perucchini".[6]

[3] Rosario Bellini era il padre di Vincenzo Bellini.

[4] Il Museo Civico Belliniano di Catania custodisce solo la seconda parte di questa lettera, la prima parte della lettera è conservata presso il Museo Correr di Venezia.

[5] Benedetto Condorelli (Catania, 1878 – Catania, 1950) fu l'incaricato dell'istituzione del Museo Belliniano di Catania.

[6] Informazioni prese dalla fonte: <https://umanisticadigitale.unibo.it/article/view/9162/9918>.

Uno dei principali destinatari della maggior parte delle lettere scritte da Bellini fu Francesco Cosimo, compagno di studi negli anni di formazione a Napoli ed amico intimo di Bellini. Il Museo attualmente, però, di queste ne custodisce solo una scritta da Bellini il 1° giugno 1835.

Bellini scriveva a Florimo regolarmente confidandogli, oltre a questioni di lavoro ed economiche, anche affari più riservati, i propri progetti, i propri stati d'animo, i propri pensieri e le proprie faccende private.

Nel 1869 Florimo pubblica due volumi in cui ricostruisce le vicende della scuola napoletana e dei suoi principali esponenti dedicando un intero capitolo a Bellini in cui compaiono ampi stralci delle sue lettere.[7] Florimo dichiara di aver avuto col compositore una lunga corrispondenza di circa otto anni e di avere conservato, di quella corrispondenza, solo una parte, riferendo che un'immensa quantità di lettere è stata donata ad « amici, ai personaggi di grandissima distinzione, ed alle gentili signore le quali ambivano di poter conservare come cosa sacra un autografo dell'autore della Norma».[8] Successivamente, nel 1882, Florimo pubblica il volume monografico che comprende anche le missive scritte da Bellini che in totale sono 101, tra cui 75 indirizzate a Florimo ed altre 26 indirizzate ad altri destinatari.[9] Queste missive, che Bellini scriveva a Florimo, risultano molto importanti perché in esse mette a nudo sé stesso e da esse vengono fuori informazioni riguardanti la società che ruota intorno al teatro d'opera, alla musica, all'arte, alla cultura del tempo.

Florimo considera l'arte dell'amico l'apice della tradizione musicale napoletana e dopo la morte di Bellini dedica la sua vita alla memoria del compositore mistificandone la figura di uomo e di artista.

Bisogna però precisare che molte lettere vennero distrutte dallo stesso Florimo per rimuovere ciò che ci poteva essere di compromettente riguardo le vicende professionali e sentimentali vissute dal musicista.[10] Florimo incaricò i maestri Daniele Napoletano e Giovanni Anfossi di fare uno spoglio accurato di tutte le lettere autografe di Bellini e di separare quelle che contenevano informazioni che si riferivano a fatti

[7] Cfr. Florimo, Francesco. 1969-71. *Cenno storico sulla scuola musicale di Napoli*. 2 voll. Napoli: Rocco.

[8] Cfr. Florimo, Francesco. 1871. *Cenno storico sulla scuola musicale di Napoli*. Vol. 2. Napoli: Rocco, p. 784.

[9] Cfr. Florimo, Francesco. 1882. *Bellini. Memorie e lettere*. Firenze: Barbèra.

[10] Si fa riferimento alle lettere spedite da Bellini a Napoli dal 14 marzo 1829 all'11 marzo 1834 ed alcune missive inviate da Parigi.

personali e privati da quelle in cui si parlava di affari, di vita musicale, di scritture ed interessi artistici. A questo punto Florimo distrusse, bruciandolo, il primo gruppo di lettere e donò la parte restante alla biblioteca del Conservatorio di San Pietro in Majella di cui nel frattempo era stato nominato direttore.

A Francesco Pastura e a Luisa Cambi si deve una radicale revisione della figura del musicista. Nel 1935 Pastura pubblica la sua raccolta *Le lettere di Bellini*[11] indicandola espressamente come la prima edizione integrale dell'epistolario del musicista.

Questa raccolta comprende, oltre alle missive già conosciute, le lettere in possesso del giurista Federico Patetta, tutte indirizzate ad Alessandro Lamperi,[12] amico di Bellini, trascritte e date alle stampe da Alessandro Luzio nel 1932 nel vol. LXVII degli *Atti della Reale Accademia delle Scienze di Torino.*[13]

Otto anni dopo, nel 1943, Luisa Cambi pubblica la prima edizione critica delle lettere di Vincenzo Bellini, edizione che ha costituito per molto tempo un punto di riferimento.[14] Questa edizione comprende le lettere scritte da Bellini, più un insieme di documenti tra cui articoli, atti, suppliche, avvisi, circolari e due lettere inerenti alla morte di Bellini, una scritta il 26 settembre 1835 da Rossini a Santocanale, e l'altra scritta il 3 ottobre 1835 da Mercandante a Florimo.

Per quanto riguarda le lettere autografe di Bellini presenti in questa edizione, sono comprese le missive conosciute fino a quel momento che comprendono le lettere pubblicate da Florimo e da Pastura più altre lettere non edite né da Florimo né da Pastura. Tra queste ne abbiamo alcune apparse nei primi contributi ottocenteschi d'impianto biografico dedicati al compositore[15] ed altre riguardanti la corrispondenza

---

[11] Pastura, Francesco. 1935. *Le lettere di Bellini (1819-1835). Prima edizione integrale raccolta, ordinata e annotata da Francesco Pastura*. Catania: Totalità.

[12] Alessandro Lamperi fu Sottosegretario di Stato al Ministero per gli Affari Esteri nel Regno di Sardegna; Bellini lo incontrò in occasione del suo viaggio a Torino nell'ottobre del 1829 e da allora tenne con lui costanti rapporti epistolari, che proseguirono sino alla morte del musicista.

[13] La raccolta comprendeva 17 lettere, di cui alcune custodite presso la Houghton Library di Cambridge in Massachusetts ed altre presso collezioni private.

[14] Cfr. Cambi, Luisa. 1943. *Bellini. Epistolario*. Milano: Mondadori.

[15] Si tratta della biografia di Bellini scritta da F. Gerardi nel 1835, quella di F. Cicconetti scritta nel 1859, la monografia *Bellini: sa vie, ses œuvres* di A. Pougin pubblicata nel 1868, il volume II delle *Ricordanze biografiche* di C. Pepoli del 1881 e la brochure contenente quattro lettere di Bellini indirizzate a Perucchini pubblicata da G. Salvioli nel 1884.

proveniente da contributi di studiosi catanesi[16] che avevano avuto modo di visionare la maggior parte degli autografi in possesso dei familiari di Bellini e di altri privati.[17]
Luisa Cambi, inoltre, esprime alcune perplessità riguardo le trascrizioni di Florimo, ritenendo di dubbia autenticità alcune lettere che, a detta di Florimo, Bellini avrebbe scritto a Londra.[18] Tali perplessità sono emerse per via che queste lettere erano state inserite nella raccolta "*Cenno storico sulla scuola musicale di Napoli"* ma non nell'edizione del 1882.
Nonostante queste esitazioni manifestate da Luisa Cambi, alcuni errori di trascrizioni, di interpretazione e «le omissioni determinate da difficoltà di decifrazione, la sua edizione appare ancora esemplare per la serietà dell'impianto, il rigore documentario e – in presenza degli autografi – la fedeltà al dettato originale».[19]
Nel 1959 Pastura, che nel 1950 era diventato direttore del Museo Civico Belliniano, pubblica una monografia[20] sul musicista nella quale trovano spazio le lettere fino a quel momento conosciute e le lettere che negli anni erano confluite nel Museo.
Si arriva così al 1973, anno in cui, la Cambi dà alle stampe un saggio[21] che promuove l'edizione integrale dell'epistolario belliniano e in cui trascrive un pacchetto di autografi inediti comprendenti lettere e documenti che le proprie ricerche,[22] quelle di Frank Walker[23] e di Friedrich Lippmann hanno portato alla luce.[24]

---

[16] Si tratta delle trascrizioni di G. Arenaprimo e F. Guardione inserite nella pubblicazione *Omaggio a Bellini nel primo centenario della sua nascita* promossa nel 1901 dal Real Circolo Bellini, delle trascrizioni di G. Libertini nel *Numero commemorativo a cura della Rivista del Comune di Catania* realizzato nel 1935 e della corrispondenza proveniente dai volumi monografici di Antonino Amore. Vedi Amore, Antonino. 1892. *Vincenzo Bellini. Arte, studi e ricerche*. Catania: Giannotta; id. 1894. *Vincenzo Bellini. Vita. Studi e ricerche*. Catania: Giannotta.

[17] Cfr. Seminara, Graziella. 2017. *Vincenzo Bellini. Carteggi*. Firenze: Olschki, pp. 12-14.

[18] Cfr. Cambi, Luisa. 1943. *Bellini. Epistolario*. Milano: Mondadori, p. 363.

[19] Seminara, Graziella. 2017. *Vincenzo Bellini. Carteggi*. Firenze: Olschki, pp. 11.

[20] Cfr. Pastura, Francesco. 1959. *Bellini secondo la storia*. Parma: Guanda.

[21] Cfr. Cambi, Luisa. 1973. *Bellini. Un pacchetto di autografi*, in *Scritti in onore di Luigi Ronga*. Milano-Napoli: Ricciardi, pp. 53-90.

[22] Si tratta di lettere conservate all'Istituto Mazziniano di Genova, in alcune biblioteche di Firenze, di Forlì, di Venezia, di Vicenza, nell'archivio di stato di Reggio nell'Emilia, a Berlino, a Vienna, a Cambridge, in Massachusetts e a New York.

[23] Cfr. Walker, Frank. 1960. «*Lettere disperse e inedite di Vincenzo Bellini*». Rivista del Comune di Catania. Vol.8, n. 4, pp. 3-15.

[24] Cfr. Lippmann, Friedrich. 1977. *Belliniana*, in *Il melodramma italiano dell'Ottocento. Studi in onore di Massimo Mila*. Torino: Einaudi, pp. 281-317.

Quasi nello stesso arco di tempo, Carmelo Neri,[25] che a partire dagli anni Novanta conduce una meticolosa ricerca bibliotecaria su tutto il territorio nazionale, trova un numero di missive di mano belliniana e arriva a curarne, assumendo come base l'edizione Cambi, due nuove edizioni.[26]

Segue poi, nel 2001, la pubblicazione della prima raccolta di lettere di cui Bellini fu il destinatario, in gran parte conservate al Museo Civico Belliniano.[27] Neri commette però l'errore di non verificare sempre il testo confrontandolo con gli autografi, finendo così per riprodurre errori ed omissioni dell'edizione Cambi pertanto venendo meno ai criteri di scientificità e rigore filologico.

Una cosa va detta: nessuno, né Pastura né Cambi né Neri avevano messo in dubbio la fedeltà delle trascrizioni di Florimo sul gruppo di lettere da lui edito. Il primo a farlo è stato John Rosselli che, nel corso del Convegno internazionale del 2000,[28] ha rivendicato la necessità che l'intero epistolario belliniano venga «vagliato con metodi di critica storiografica già messi in opera nel Quattrocento da Lorenzo Valla»,[29] e ha sottoposto le lettere trascritte da Florimo a una rilettura critica arrivando a sollevare dubbi addirittura sulla loro autenticità. Florimo, per interpretare le lettere scritte da Bellini, molte delle quali rovinate, si rivolse al suo amico Cesare Dalbono, a cui affidò l'incarico di trascriverle.

Su queste, poi, Florimo intervenne apportando correzioni sul piano fonomorfologico, sintattico e lessicale «con lo scopo di consegnare una versione stilisticamente affinata della scrittura belliniana, che nei manoscritti originali si rivela al contrario fortemente spontanea».[30]Secondo Rosselli Florimo però, oltre che sullo stile epistolare del

---

[25] Carmelo Neri, laureato nel 1968 in Lettere Moderne presso l'Università di Messina, è autore di numerose pubblicazioni su varie riviste e nei programmi di sala del Teatro Massimo di Catania. Ha svolto lunghe ed appassionate ricerche su Bellini.
Cfr. <https://www.algraeditore.it/autore/carmelo-neri/>.

[26] Cfr. Neri, Carmelo. 1991. *Lettere di Vincenzo Bellini. 1819-1835*. Catania: Publisicula;
2005. *Vincenzo Bellini. Nuovo Epistolario. 1819-1835*. Catania: Agorà.

[27] Neri, Carmelo. 2001. *Caro Bellini... Lettere edite e inedite a Vincenzo Bellini*. Catania: Prova d'autore.

[28] Convegno "*Vincenzo Bellini. Verso l'edizione critica"* tenuto all'Accademia Chigiana di Siena nel giugno 2000.

[29] Rosselli, John. 2004. «Per *un'edizione critica dell'epistolario belliniano*, in *Vincenzo Bellini. Verso l'edizione critica*». Atti del Convegno internazionale (Siena, 1-3 giugno 2000), a cura di Fabrizio Della Seta e Simonetta Ricciardi. Firenze: Olschki, p. 292.
Cfr. Seminara, Graziella. 2017. *Vincenzo Bellini. Carteggi*. Firenze: Olschki, p. 8.

[30] Seminara, Graziella. 2017. *Vincenzo Bellini. Carteggi*. Firenze: Olschki, pp. 13.

musicista, agì anche sul contenuto sostanziale dei testi manipolando e falsificando le epistole in suo possesso per restituire un profilo elevato all'artista.

In occasione del bicentenario dalla nascita di Bellini il comitato delle celebrazioni nazionali ha sottolineato l'esigenza di preparare una nuova edizione del carteggio avente come scopo non solo quello di riunire tutti i testi sparsi e di mettere nuovi ritrovamenti a disposizione della comunità scientifica, ma anche quello di verificare i testi per valutarne l'autenticità o la contraffazione offrendo versioni che rispecchiano validi criteri linguistici e filologici.[31]

Tutti gli studi, resi possibili dal Centro di Documentazione per gli Studi Belliniani dell'Università degli Studi di Catania, hanno trovato concretizzazione nell'edizione critica curata da Graziella Seminara e uscita nel 2017.[32]

Quest'edizione è il punto di riferimento per la creazione dell'edizione digitale delle lettere scritte da Bellini che sono conservate presso il Museo Civico Belliniano di Catania:[33] è interessate che tale edizione, consultabile all'interno del percorso museale, è indirizzata anche a utenti con diversa preparazione musicale.

Il *corpus* presente nell'edizione di Seminara è composto da 517 documenti, ordinati secondo un principio cronologico, sono cioè testi che vanno dalla supplica scritta al duca Sammartino il 3 maggio 1819, ai biglietti scritti da Bellini nei suoi ultimi giorni di vita, l'ultimo di questi scritto il 18 settembre 1835. Per ogni lettera, oltre alla trascrizione, vengono fornite informazioni riguardanti il destinatario, il luogo e la data di compilazione, l'attuale ubicazione dell'autografo e la consistenza del documento. Inoltre, in aggiunta alla trascrizione troviamo anche le note esplicative che contengono le informazioni sui personaggi e sugli eventi citati, sui rapporti di questi con Bellini e molti particolari utili per contestualizzare e comprendere meglio il documento. Tutto questo materiale è utile anche per documentare la meticolosa contabilità e l'istinto affaristico con cui Bellini curava i propri interessi economici. Il lavoro editoriale è eseguito dalla Seminara con cura e precisione e l'edizione è ricca di informazioni e presenta serietà di controllo filologico sugli originali.[34]

---

[31] Ibidem.

[32] Cfr. Seminara, Graziella. 2017. *Vincenzo Bellini. Carteggi*. Firenze: Olschki.

[33] Cfr. <http://bellinicorrespondence.cnr.it>.

[34] Cfr. Coletti, Vittorio. 2017. «*Vincenzo Bellini – Carteggi. Silenzio universale, applausi unanimi,* L'Indice». Vol. 10, p. 17.
Cfr. <https://www.lindiceonline.com/l-indice/sommario/ottobre-2017/>.

2. Definizione del progetto *Bellini Digital Correspondence*

2.1. *Bellini Digital Correspondence* (BDC)

Bellini Digital Correspondence si inserisce nel contesto più ampio del progetto *Museo Virtuale della Musica BellinInRete*.[35] Si tratta di un'iniziativa altamente multidisciplinare che coinvolge varie istituzioni tra cui: l'Università di Catania, l'Istituto di Scienze e Tecnologie della Cognizione (ISTC) del CNR di Catania e l'Istituto di Linguistica Computazionale "A. Zampolli" (ILC) del CNR di Pisa.

Il progetto *BellinInRete* si concentra sull'analisi, sull'organizzazione e sul rinnovamento della fruizione del patrimonio del Museo Civico Belliniano di Catania in cui sono conservate le lettere autografe del compositore catanese.

Per coinvolgere i visitatori del museo è stato previsto un allestimento scenografico che evoca i vari ambienti del teatro dell'epoca consentendo un'immersione totale da parte del visitatore. Per la fruizione a distanza invece è stato previsto un museo digitale multicanale che espone on line i contenuti dei documenti con modalità interattive.

Il progetto *Bellini Digital Correspondence* nello specifico si concentra sullo sviluppo e sulla fruizione dell'edizione scientifica digitale delle lettere autografe di Vincenzo Bellini (conservate a Catania) e ne comprende la trascrizione, la codifica, la gestione, l'indicizzazione e la resa a video.[36]

Le carte (lettere, partiture, documenti archivistici) custodite presso il museo sono state digitalizzate e in quest'ottica si inserisce come edizione scientifica digitale image-based (mediante la giustapposizione del testo con la corrispondente scansione dell'originale).

Le lettere che compongono il repertorio sono state trascritte sulla base dell'edizione critica a stampa a cura di Graziella Seminara e sono state ulteriormente arricchite di dati strutturali, funzionali e semantici grazie all'adozione delle linee guida TEI.

La fruizione dell'edizione digitale invece avviene mediante l'uso del software di visualizzazione web EVT così da garantire una presentazione completa sia dell'immagine della fonte primaria, sia della sua trascrizione nonché delle

---

Cfr. <https://www.lindiceonline.com/focus/musica/vincenzo-bellini-carteggi/>.

[35] <http://bellininrete.istc.cnr.it/>.

[36] L'edizione è consultabile sui siti: <http://bellinicorrespondence.cnr.it/> e <http://licodemo.ilc.cnr.it/bellini-in-rete>.

informazioni dettagliate sul *corpus* belliniano allo scopo di rendere accessibili le risorse a un vasto pubblico.

2.2. TEI (Text Encoding Initiative)

La Text Encoding Initiative (TEI) è un consorzio internazionale composto da istituzioni accademiche, studiosi e progetti di ricerca.
L'obiettivo di questa organizzazione è lo sviluppo e il mantenimento di uno schema di codifica internazionalmente conosciuto per la rappresentazione scientificamente curata dei testi in forma digitale. A tal proposito la TEI ha redatto una serie di linee guida che specificano i metodi di codifica per testi in formato digitale nel campo delle scienze umane, sociali e della linguistica. Le TEI Guidelines[37] sono state ampiamente utilizzate da biblioteche, musei, editori e singoli studiosi al fine di digitalizzare i testi per la ricerca, per l'insegnamento e per la conservazione.
Le linee guida raccomandano un metodo rigoroso e formale per la rappresentazione di risorse testuali specificando un insieme di etichette metatestuali (chiamati anche tag o marcatori) che concorrono alla rappresentazione elettronica del testo per marcarne la struttura ed esplicitarne le altre caratteristiche di interesse.
L'insieme delle regole stabilite associate a questo procedimento di marcatura si definisce *linguaggio di markup* o *codifica* mentre nel *vocabolario di markup* sono invece compresi l'insieme dei marcatori (detti anche tag oppure elementi) definiti da uno schema di codifica.
Il linguaggio di markup definito dalla TEI è espresso da uno schema formale di codifica XML e permette di rappresentare le caratteristiche strutturali e concettuali dei testi letterari. Per queste ragioni lo schema XML messo a punto dalle TEI è considerato uno standard de facto per la codifica scientifica dei testi.
L'impiego delle TEI Guidelines presenta numerosi vantaggi:[38]

- sono semplici, chiare e concrete;
- sono di facile utilizzo e non richiedono l'utilizzo di software specializzati;
- si possono applicare a testi di qualsiasi lingua, di qualsiasi epoca e di qualsiasi genere letterario o tipo di testo, senza vincoli su forma o contenuto.

[37] <https://tei-c.org/>.

[38] <https://it.wikipedia.org/wiki/Text_Encoding_Initiative>.

La versione più recente delle linee guida è la TEI P5[39] pubblicata il 1º novembre 2007. Le norme, le raccomandazioni e il vocabolario sono espresse con il linguaggio di marcatura più diffuso per le risorse digitali: l'Extensible Markup Language (XML).
Gli elementi principali di un documento TEI sono la radice `TEI` e gli elementi `teiHeader` e `text`. L'elemento `teiHeader`, l'intestazione cioè del documento, fornisce metadati descrittivi e dichiarativi associati a una risorsa digitale. All'interno dell'elemento `teiHeader` sono compresi gli elementi `fileDesc` (elemento obbligatorio) e ulteriori elementi non obbligatori quali `profileDesc`, `encodingDesc`, `revisionDesc`. L'elemento `text` rappresenta la trascrizione del testo il cui modello di contenuto comprende gli elementi `front`, `body` e `back`. L'elemento `body` può includere una serie di suddivisioni ulteriori, necessarie a definire la struttura interna del testo (come, per esempio, gli elementi `div` e `p`).[40]

2.3. XML (Extensible Markup Language)

XML[41] è un metalinguaggio[42] sviluppato e mantenuto dal W3C[43] (World Wide Web Consortium) attraverso il quale implementare linguaggi di markup descrittivi. Il linguaggio XML è utilizzato per definire metodi di elaborazione e di archiviazione di documenti testuali in formato elettronico. L'eredità di XML risiede nel precedente metalinguaggio SGML[44] (Standard Generalized Markup Language) creato da Goldfarb negli anni '70 con l'obiettivo di definire linguaggi da usare per la stesura di testi destinati ad essere archiviati e trasmessi tramite strumenti informatici. SGML ha

[39] <https://tei-c.org/guidelines/p5/>.

[40] <https://tei-c.org/release/doc/tei-p5-doc/en/html/CO.html>.

[41] <https://tei-c.org/release/doc/tei-p5-doc/en/html/SG.html>.

[42] Per *metalinguaggio* si intende un tipo di linguaggio che segue un insieme di regole sintattiche con lo scopo di definire e descrivere linguaggi artificiali.
<https://it.wikipedia.org/wiki/Metalinguaggio>.

[43] Il W3C è un'organizzazione non governativa internazionale che ha come scopo il potenziamento continuo del World Wide Web.
<https://www.w3.org/>.

[44] Doug, Tidwell. 2008. *XSLT: mastering XML transformation.* United States of America: O'Reilly Media.
<https://it.wikipedia.org/wiki/Standard_Generalized_Markup_Language>.

ottenuto una grande diffusione quando Tim Berners-Lee ha basato l'HTML sulle specifiche richieste da SGML.

Lo XML viene spesso messo in relazione con l'HTML.

Rispetto all'HTML, tuttavia, XML ha alcune importanti differenze:

- come riportato nel suo stesso nome, è *eXtensible* ovvero estensibile. Questo vuol dire che non presenta un insieme fisso di tag e che si può personalizzare in base a specifiche esigenze;
- pone l'accento sulla marcatura descrittiva piuttosto che procedurale e quindi è adatto alla rappresentazione del contenuto semantico piuttosto che della resa grafica del testo;
- distingue i concetti di correttezza sintattica e di *validità* rispetto ad una *DTD (Document Type Definition)*;
- è indipendente da qualsiasi sistema hardware o software.

XML, inoltre, si basa su un modello di dati di tipo gerarchico con una struttura ad albero etichettato ordinato.[45] La struttura ad albero principale di un documento XML è formata da nodi rappresentati attraverso dei marcatori chiamati *tag* racchiusi tra parentesi angolari.

[45] Una *struttura dati* è un'entità usata per rappresentare in modo efficiente un insieme di dati all'interno della memoria di un computer; si tratta, quindi, di un metodo di organizzazione dei dati.
Il *modello gerarchico*, su cui si basa la struttura dei documenti XML, prevede che i dati siano organizzati secondo strutture ad albero che riflettano una gerarchia presente tra le entità. Questo modello consente di rappresentare la relazione tra i nodi padre figli.
La *struttura ad albero* è una struttura dati logica dove ogni elemento è collocato all'interno di un ordine gerarchico. Un albero si compone di due sottostrutture fondamentali: il nodo, che contiene l'informazione, e l'arco che stabilisce un collegamento gerarchico tra due nodi. Ogni nodo può essere collegato con altri nodi a livello gerarchico inferiore. Il nodo denominato *padre* è quello da cui origina l'arco orientato che lo collega a un nodo figlio che si trova a livello gerarchico inferiore denominato *figlio*. La radice (root) è il nodo di ordine gerarchico più alto (e non presenta un nodo padre).
Un albero può essere *ordinato* o *non ordinato*. Si parla di *albero non ordinato* nel caso in cui non segua nessuna regola per quanto riguarda la relazione d'ordine padre-figlio.
Un albero è invece *ordinato* se è l'ordinamento tra il nodo padre e i nodi figli è significativo.
Un albero può essere *etichettato* se è costituito da nodi rappresentati da un nome o che veicolano un'informazione semantica sulla base di un'etichetta associata. I tag e gli elementi possono presentare inoltre degli attributi che specificano ulteriormente il nodo definendo nuove informazioni.

Per approfondimenti si rimanda a risorse bibliografiche quali:
Sedgewick, Robert, e Kevin Wayne. 2011. *Algorithms*. 4th Edition. United States of America: Addison-Wesley Professional. https://algs4.cs.princeton.edu/home/.
Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, e Clifford Stein. 2022. *Introduction to Algorithms*. Fourth Edition. United States of America: The MIT Press.

2.4. EVT: Edition Visualization Technology

EVT[46] è un software open-source basato sugli standard del web (HTML, CSS, Javascript), sviluppato per pubblicare edizioni scientifiche digitali di testi codificati secondo gli schemi e le linee guida TEI XML. Il viewer consente agli utenti la navigazione, l'esplorazione e lo studio delle edizioni digitali attraverso un'interfaccia di semplice utilizzo.

Nato nell'ambito del progetto *Digital Vercelli Book*,[47] si è evoluto in uno strumento in grado di adattarsi a differenti testi ed esigenze. Il software si presenta con un set completo di strumenti flessibili ed è personalizzabile sviluppato per consentire agli utenti di visualizzare, leggere e confrontare le edizioni in un ambiente elettronico.

EVT presenta un layout estensibile, ridimensionabile, con capacità di nidificazione e compatibilità con i maggiori *framework* Javascript per le interfacce GUI.[48] In aggiunta il software presenta immagini e testo in frame separati, un collegamento avanzato immagine-testo, la navigazione per miniature delle immagini, uno zoom in-out e una lente di ingrandimento.

3. Sviluppo del progetto BDC

La struttura del *corpus* epistolare belliniano, a cui fa riferimento il progetto BDC, si compone di 40 unità testuali che sono state riprodotte in formato facsimilare per mezzo di scansioni. Le immagini riproducono le singole carte (recto, verso).

All'interno del carteggio belliniano la segnatura LL1 corrisponde alle missive scritte da Vincenzo Bellini. La lettera n-esima in ordine cronologico si indica con LL1.*n*. Il numero I o II, in coda alla segnatura (quando presente), specifica la coesistenza di più missive nella stessa carta (unità testuale).

Lo sviluppo del progetto BDC si compone di diverse fasi redazionali tra cui:

- la selezione dei materiali per lo sviluppo dell'edizione;
- la codifica delle lettere del *corpus* epistolare in formato XML seguendo le linee guida TEI;

---

[46] <http://www.labcd.unipi.it/progetti/evt-edition-visualization-technology/>.

[47] <http://vbd.humnet.unipi.it/>.

[48] <https://it.wikipedia.org/wiki/Interfaccia_grafica>.

- la validazione formale e scientifica dei documenti codificati;
- l'armonizzazione delle diverse codifiche;
- la creazione del *corpus* belliniano;
- la fruizione e la presentazione dell'edizione.

3.1. Selezione dei materiali

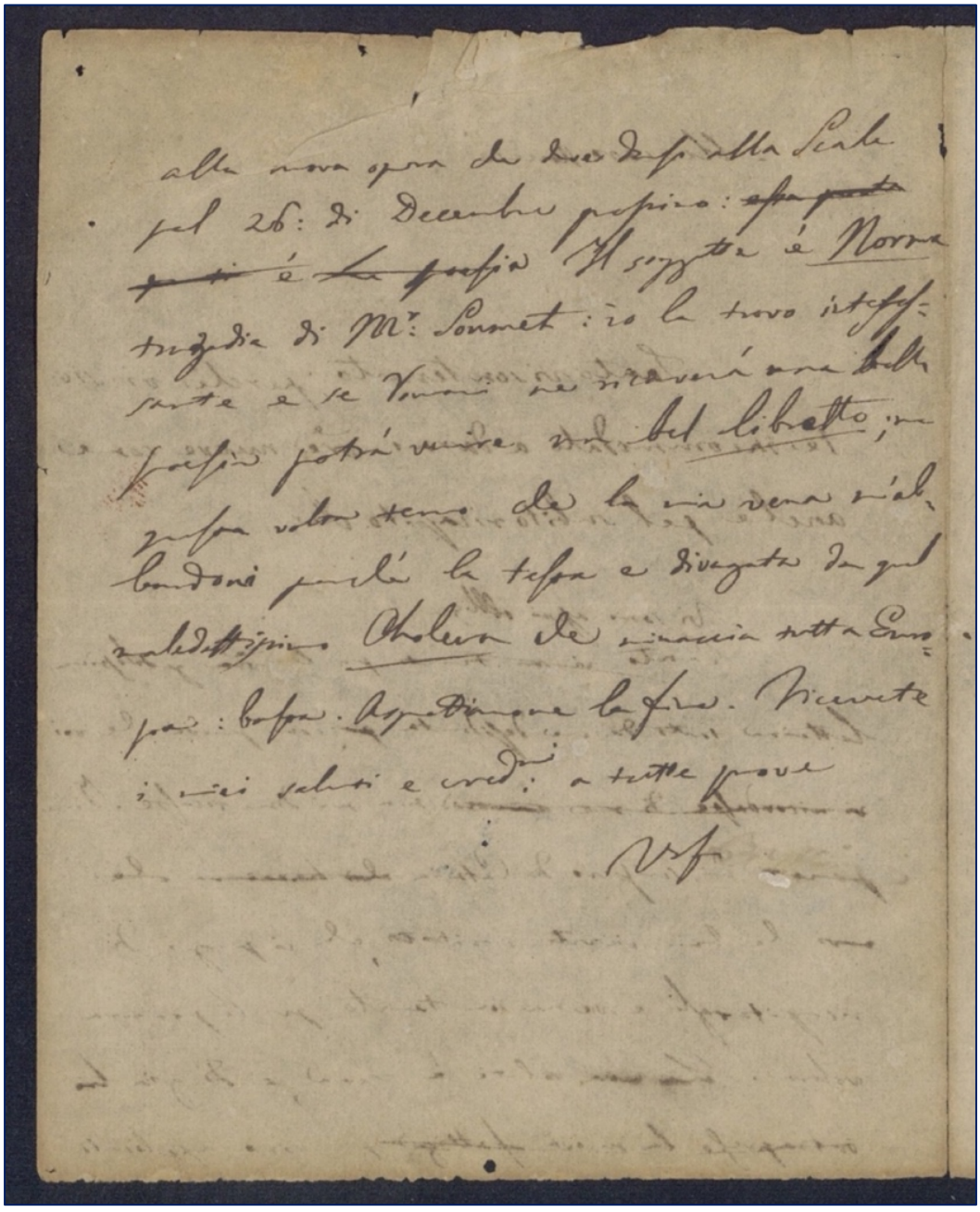

Figura 1. Lettera LL1.8.

Figura 2. Imbustamento della lettera LL1.9.

Per lo sviluppo dell'edizione sono stati selezionati i materiali da utilizzare allo scopo. Nello specifico, per la realizzazione dei documenti digitali in formato XML, sono stati messi a disposizione materiali di tipo archivistico e bibliografico tra cui: un modello di codifica per ogni lettera, la trascrizione dei testi derivata dall'edizione della Seminara, le immagini (vedi ad esempio le figure 1 e 2), e le liste delle persone, dei luoghi, delle organizzazioni, dei termini, delle opere e della bibliografia. Inoltre, è stato stilato un elenco dettagliato degli elementi e degli attributi con i relativi valori, per la corretta codifica delle missive, organizzato in documenti di lavoro pubblicati sul repository GitHub[49] di progetto, in modo da rendere il più efficiente possibile le attività del gruppo di ricerca.

Entità citate codificate nell'edizione digitale BDC	
Luoghi	23
Organizzazioni	11
Opere	12
Termini	25
Persone	77
Entrate bibliografiche	14

3.2. Codifica delle lettere

Il progetto BDC si inserisce in un contesto didattico che prevede la codifica di buona parte del *corpus* da parte di tirocinanti, tesisti e studenti del corso di codifica dei testi dell'Università di Pisa.

Il repertorio prodotto è costituito da 40 documenti XML riguardanti le unità testuali e da 6 documenti XML a corredo che invece contengono le liste delle entità nominate e di altri dati rilevanti, tra cui termini musicali (come ad esempio *aria*, *soprano*, *duetto*, *partitura*, *quintetto*, *tenore*, ecc.) con il loro significato ed opere (tra cui *La sonnambula*, *I puritani*, *Il pirata*, *Norma*, *I Capuleti e i Montecchi*, *La straniera*, ecc.) con i relativi approfondimenti su di esse, che sono menzionati nelle lettere.

[49] <https://github.com/>.

Il processo di codifica ha coinvolto diverse decine di studenti che hanno selezionato e annotato le lettere del *corpus* belliniano.

La codifica è stata realizzata in formato XML seguendo le linee guida TEI e la visualizzazione delle singole lettere è avvenuta in modo autonomo con risultati molto diversi tra loro attraverso la realizzazione di proposte di edizione digitale. Per questa ragione la codifica dei documenti del *corpus* inizialmente non era uniforme e ha necessitato di un attento processo di armonizzazione per la realizzazione dell'edizione digitale finale.

I frammenti XML principali dei documenti di codifica sono: l'elemento radice `TEI` e gli elementi figli `teiHeader`, `facsimile` e `text`.

L'elemento `teiHeader` contiene l'elemento `fileDesc` che raggruppa gli elementi che descrivono il documento; qui sono codificate le informazioni riguardanti il titolo, l'autore, i responsabili della codifica della lettera, la fonte da cui il documento digitale è derivato, le caratteristiche del manoscritto codificato, la sua collocazione, la sua condizione fisica e l'eventuale presenza di timbri, sigilli, filigrane, segni di usura e di grafie appartenenti a mani diverse da quelle di Bellini.

L'elemento `facsimile` codifica la rappresentazione del manoscritto sotto forma di immagini. L'elemento `text` codifica il testo della lettera ed è composto da tre elementi: `front`, `body` e `back`. L'elemento `front` codifica i dati relativi all'imbustamento della lettera (nome ed indirizzo del mittente e del destinatario, data e luogo di spedizione, timbri, ecc.), l'elemento `body` codifica il corpo del testo (tra cui le righe codificate con l'elemento `lb`) e l'elemento `back` comprende note esplicative (elemento `note`) e riferimenti bibliografici (elemento `bibl`) di altre edizioni critiche della missiva.

3.3. Armonizzazione della codifica

La fase di armonizzazione ha previsto la revisione e la registrazione di fenomeni testuali nel documento digitale tra cui:

- la codifica accurata dei dati relativi alla corrispondenza, della componente degli indirizzi e dei dati riguardanti l'imbustamento;

- l’inserimento e la revisione della codifica dell’intestazione di ogni lettera, delle note, della bibliografia, dell’interruzione di linea, delle cancellature e delle annotazioni di altre mani;
- la codifica e la revisione delle regioni d’interesse a partire dalle fonti facsimilari;
- lo scioglimento delle abbreviazioni;
- la codifica dei collegamenti e dei riferimenti alle entità esterne;
- il completamento e la revisione delle descrizioni del supporto scrittorio e della conformazione fisica delle lettere con eventuale codifica di timbri, segni, danni, strappi, sigilli, piegature, dello stato di conservazione ed altri fenomeni materiali.

Nelle figure 1 e 2, ad esempio, si possono visualizzare alcuni fenomeni codificati tra cui: cancellature, sottolineature, andata a capo, abbreviazioni, correzioni, l’imbustamento, i timbri e sigilli.

3.4. Presentazione e fruizione dell’edizione

La presentazione e la fruizione dell’edizione digitale si ottiene grazie alla pubblicazione in rete dei dati attraverso la tecnologia web con l’utilizzo del software EVT (v. figura 3).

Per elaborare correttamente la codifica XML, EVT legge in input un singolo documento TEI con specifiche modalità di compilazione. Per questo si è reso indispensabile realizzare l’intero *corpus* attraverso un meccanismo di inclusione dinamica dei singoli documenti che si basa sulla tecnologia XSLT garantendo modularità, manutenibilità e flessibilità.

In più, EVT richiede una struttura del documento TEI differente dal modello di codifica adottato per le lettere belliniane, quali, ad esempio, le informazioni bibliografiche, la gestione dei metadati, le descrizioni codicologiche, la gestione dei dati facsimilari, le note, la terminologia.

Per pubblicare tutti i dati disponibili sono state implementate opportune trasformazioni strutturali dei documenti XML del repertorio che verranno illustrate puntualmente nei capitoli successivi.

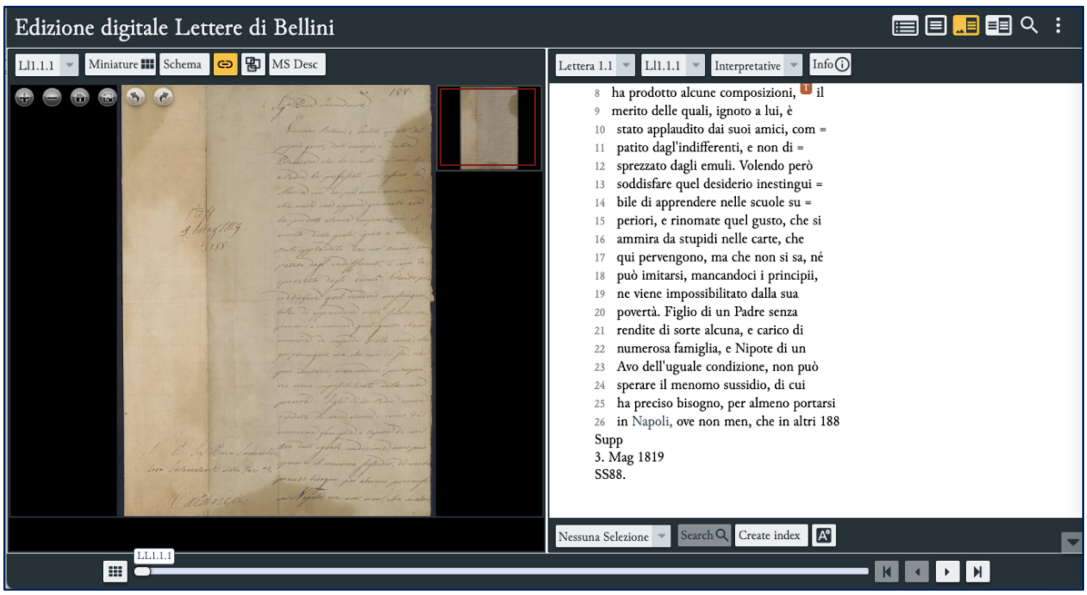

Figura 3. Edizione digitale *Bellini Digital Correspondence* su EVT.

4. Processo di trasformazione mediante XSLT

4.1. XSLT: eXtensible Stylesheet Language Transformations

XSLT[50] (acronimo di eXtensible Stylesheet Language Transformations) è un linguaggio dichiarativo-funzionale che permette la manipolazione e la trasformazione di documenti XML grazie all'applicazione di istruzioni XSL.[51]

La principale funzione di XSLT è quella di trasformare la struttura e il formato di un documento XML (di input) in base alle istruzioni definite in un foglio di stile XSLT così che un'unica fonte o un insieme di documenti di origine possano essere riusati per produrre output in diversi formati (per esempio nello stesso formato XML con struttura e contenuto diverso oppure in formati differenti come XHTML, testo, RTF, PDF, ecc.). XSLT impiega un ulteriore linguaggio, XPath,[52] con cui, attraverso l'uso di

---

[50] <https://www.w3.org/TR/xslt-30/>.
Per approfondimenti si rimanda a risorse bibliografiche quali:
Doug, Tidwell. 2008. *XSLT: mastering XML transformation.* United States of America: O'Reilly Media.

[51] XSL (acronimo di eXtensible Stylesheet Language) è il linguaggio di descrizione dei fogli di stile per i documenti in formato XML.

[52] XPath è un linguaggio di definizione di espressioni di percorso per selezionare nodi o un insieme di nodi in un documento XML.
<https://www.w3schools.com/xml/xpath_intro.asp>.

espressioni, è possibile individuare i nodi del documento XML su cui applicare le regole di trasformazione.

L'insieme delle istruzioni da eseguire per applicare le trasformazioni sono contenute in *template*s e vengono applicate su specifici nodi del documento, identificati tramite espressioni XPath.

Il processore XSLT è un software in grado di interpretare le istruzioni XSLT contenute nel foglio di stile. Il programma segue la struttura gerarchica del documento di input, individua ogni nodo grazie alle istruzioni presenti nel foglio di stile ed applica le regole del template stabilite.

Attualmente esistono tre versioni di XSLT.[53] XSLT 1.0 è stato definito dal World Wide Web Consortium (W3C) nel 1999. Una delle prime implementazioni delle specifiche XSLT 1.0 è stato il progetto Saxon a cui poi ne seguirono altre. Nel 2007 segue lo sviluppo di XSLT 2.0 e Saxon ha contribuito all'implementazione dello standard; questo avanzamento offre immensi miglioramenti in termini di funzionalità e produttività rispetto alla versione precedente. Nel 2012 Saxonica[54] produce Saxon-CE, la prima implementazione di XSLT 2.0 che non solo è in grado di funzionare nel browser ma che è in grado anche di gestire l'interazione dell'utente.

Le specifiche per XSLT 2.0 hanno raggiunto lo stato di raccomandazione W3C nel gennaio 2007, dopo un lungo processo di sviluppo. XSLT 2.0 ha aggiunto potenti funzionalità per la gestione del testo e dei dati strutturati aumentando in questo modo la gamma di attività a cui è possibile applicare XSLT.

Da quando XSLT 2.0 è stato finalizzato nel 2007, il W3C Working Group ha rivolto la sua attenzione allo sviluppo di XSLT 3.0 il cui avanzamento pone l'attenzione sulle trasformazioni in streaming.

4.2. Processore XSLT Saxon

Saxonb-XSLT[55] è uno strumento a riga di comando sviluppato dalla società Saxonica che permette al processore XSLT 2 l'elaborazione delle istruzioni indicate nei template presenti nel foglio di stile. Il comando che esegue il processore è invocato da riga di

[53] <https://www.saxonica.com/technology/xslt-and-xquery.xml#xslt>.

[54] <https://www.saxonica.com/>.

[55] <http://saxon.sourceforge.net/>.

comando e ha come parametri il documento XML di input, il foglio di stile e viene indicato anche il nome del file di output.
Un esempio di comando dal terminale invocando l'eseguibile dal pacchetto java jar:
`java -jar saxon-he-10.3.jar -s:sourcefile.xml -xsl:stylesheet.xsl > output.xml.`

Un esempio di riga di comando dal terminale invocando il comando installato su sistema operativo Windows o Linux:
`saxonb-xslt -s:sourcefile.xml -xsl: stylesheet.xsl -o: output.xml.`

Quando si avvia, il processore legge il documento di input e lo rappresenta in un insieme di nodi che hanno la struttura di un albero etichettato e ordinato. Ogni nodo viene navigato tramite X-Path e confrontato con le istruzioni presenti nel foglio di stile XSLT. Vengono poi eseguite le regole definite nei templates creando così l'albero di output corrispondente trasformando il file di input iniziale e generando il nuovo documento XML di output che sarà utilizzato per le fasi successive del processo.

4.3. Funzionamento del processo di trasformazione

Un foglio di stile XSL è costituito da uno o più insiemi di regole chiamate modelli o templates. L'elemento `xslt:template` viene utilizzato per creare modelli e contiene regole da applicare quando viene attraversato un nodo specificato.
L'attributo `@match` dell'elemento `xslt:template` viene utilizzato per associare un modello ad un nodo del documento di input (un nodo può essere un elemento, un attributo, il documento,…) e il suo valore è costituito da un'espressione XPath. Questo attributo può essere utilizzato anche per definire un modello per l'intero documento XML.[56]
Per eseguire la trasformazione del documento XML in cui è codificata ogni lettera si esegue il processore Saxonb-XSLT che interpreta il foglio di stile XSLT e produce il documento XML di output da utilizzare con EVT.
Per eseguire la trasformazione della lettera *LL1_1.xml*, ad esempio, applicando il foglio di stile *bellini.xsl* si deve eseguire il seguente comando:

[56] L'attributo `@match="/"` associa il modello alla radice del documento sorgente XML e definisce l'intero documento.

```
java -jar saxon-he-10.3.jar -s:LL1_1.xml -xsl:bellini.xsl
> LL1_1T.xml
```

Lo stesso procedimento va seguito per tutti i restanti documenti XML di tutte le lettere che appartengono al *corpus*. Otterremo quindi i nuovi documenti XML di output da utilizzare per EVT.

4.4. Uso dell'istruzione XInclude

Per ottenere il documento finale, che comprende tutte le lettere del *corpus* da usare con EVT, occorre integrare le varie porzioni di codice XML dai documenti prodotti con Saxonb-XSLT.

Per conseguire ciò si utilizza il meccanismo XInclude[57] creando un documento XML, chiamato *source.xml*, che consente di riunire le porzioni di codice XML da copiare dai vari documenti che sono stati ottenuti con le trasformazioni (v. figura 4).

Sono stati inclusi i blocchi di codice che riguardano gli elementi:

- `msitem` inserito nell'elemento `msContents` contenuto nel percorso `teiHeader/fileDesc/sourceDesc/msDesc`;
- `ListPerson`, `ListPlace` e `ListOrg` inseriti nell'elemento `fileDesc` contenuto in `teiHeader`;
- `facsimile;`
- `text;`

XInclude è un meccanismo di inclusione che consente di unire documenti XML o parte di essi utile per ottenere modularità nei documenti e servire efficientemente applicazioni che necessitano di documenti di grandi dimensioni. La sintassi sfrutta i costrutti XML esistenti: attributi, elementi e riferimenti URI.

Nello specifico vengono usati gli elementi `xi:include` e `xi:fallback`. L'elemento `xi:include` contiene l'attributo `@href` che specifica la posizione della risorsa da includere e l'attributo `@xpointer` utilizzato per identificare il frammento della risorsa da includere. L'elemento `xi:fallback` è contenuto nell'elemento `xi:include` e fornisce un meccanismo di ripristino delle risorse mancanti.

[57] <https://www.w3.org/TR/xinclude-11/>.

Il contenuto presente nell'elemento `xi:fallback` viene inserito nel caso in cui non è possibile risolvere l'inclusione e si verifica un errore per la risorsa indicata dall'elemento `xi:include`.

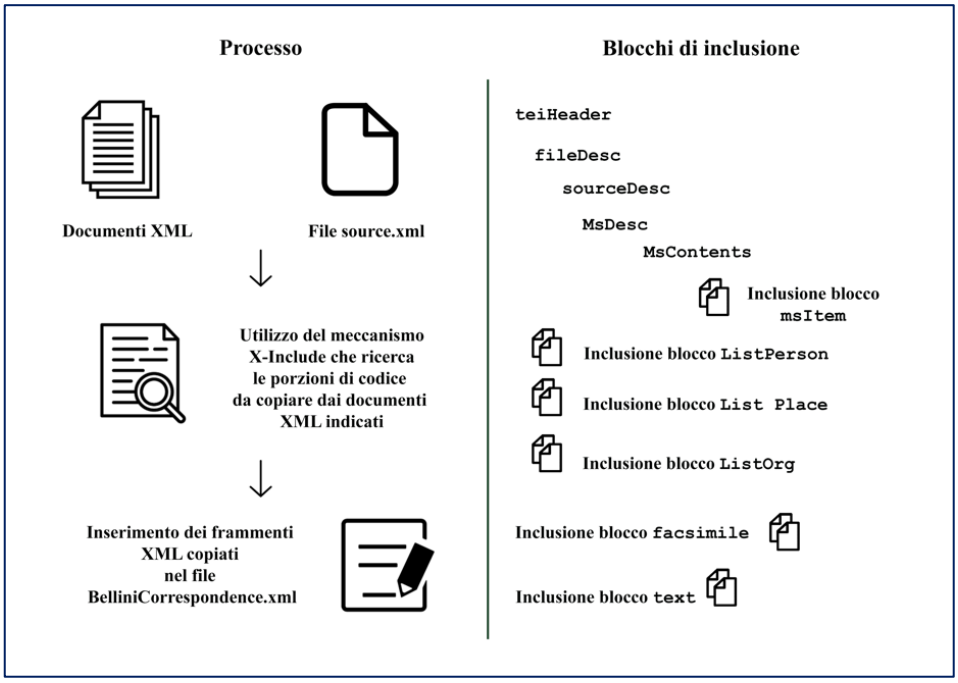

Figura 4. Processo e blocchi di inclusione.

Il codice che segue permette di copiare gli elementi `msitem` nell'elemento `msContents`:

```
<xi:include href="BelliniLetters/LL1_1T.xml"
xpointer="msItem-LL1.1"
xmlns:xi="http://www.w3.org/2001/XInclude">
  <xi:fallback><p>Contenuto non trovato</p></xi:fallback>
</xi:include>
[…]
```

Il codice che segue permette di copiare invece gli elementi `facsimile`:

```
<xi:include href="BelliniLetters/LL1_1T.xml"
xpointer="LL1.1_facsimile"
xmlns:xi="http://www.w3.org/2001/XInclude">
[…]
```

Il codice che segue permette di copiare anche gli elementi `text`:

```
<xi:include href="BelliniLetters/LL1_1T.xml"
xpointer="LL1.1"
xmlns:xi="http://www.w3.org/2001/XInclude">[...]
```

Una volta creato il file di struttura utilizzato con il meccanismo XInclude per poterlo eseguire, si possono usare strumenti quali XmlLint[58] attraverso la riga di comando che segue:

```
xmllint --xinclude --format --output
BelliniCorrespondence.xml source.xml
```

L'opzione `--xinclude` esegue l'elaborazione XInclude, l'opzione `--format` riformatta e reindenta l'output, l'opzione `--output` definisce un percorso in cui XmlLint salverà il risultato dell'analisi ed infine il file *source.xml* è quello che contiene i meccanismi di inclusione che deve elaborare XInclude.
Si ottiene così il file di output BelliniCorrespondence.xml che è il file finale utilizzato su EVT.

5. Stato dell'arte

5.1. Metodo e tecniche di ricerca

Le attività del progetto di tesi[59] sono state svolte nell'ambito del progetto *Bellini Digital Correspondence* seguendo le attività proposte dal progetto.
Tra le varie mansioni svolte il lavoro si è concentrato prevalentemente sull'analisi e sulla revisione della codifica del *corpus* belliniano e sull'applicazione delle armonizzazioni necessarie. È seguito poi un attento studio e analisi delle regole XSLT inizialmente proposte per la realizzazione delle trasformate.

---

[58] *XmlLint* consente di elaborare, navigare, validare, ecc. dei file di XML tramite riga di comando. Permette inoltre di analizzare uno o più file XML, stampare vari tipi di output, a seconda delle opzioni selezionate ed è utile per rilevare errori.
<http://xmlsoft.org/xmllint.html>.

[59] Alcune di queste attività sono state svolte in seno al periodo di tirocinio curriculare, come ad esempio l'armonizzazione dei documenti di codifica del *corpus*.

Infine sono state apportate delle modifiche per alcuni template in modo da ottenere una trasformata più completa che permettesse la visualizzazione in maniera ottimale delle informazioni desiderate.
Il lavoro di ricerca della tesi è continuato concentrandosi sulla presentazione e sulla fruizione dell'edizione digitale. Nello specifico l'esperienza dell'utente nell'ambiente web è stata ottimizzata attraverso un processo di adattamento del *corpus* con un'attività di ricerca sulle trasformazioni da effettuare per ottenere tale obiettivo.
In particolare, è stata condotta un'attenta analisi sul repertorio di lettere codificato, finalizzata all'implementazione ottimale della struttura del *corpus* per un'adeguata fruizione con il software EVT.

5.2. Fasi del lavoro

Il processo di ricerca si suddivide in diverse fasi:

- analisi del *corpus* e dell'interfaccia di EVT;
- scelta delle informazioni di interesse da visualizzare con il viewer;
- studio delle trasformazioni da applicare;
- implementazione delle regole di trasformazione del foglio di stile XSLT;
- elaborazione XSLT;
- generazione del file di output finale per la corretta visualizzazione.

La prima fase ha previsto l'analisi del *corpus* belliniano e dell'interfaccia di EVT per la fruizione dell'edizione su di esso. È stato analizzato il *corpus* per capire quali dati riportare e quali invece sono superflui per la fruizione ed è stata analizzata l'interfaccia web di EVT per capire quali informazioni sono elaborabili e come vengono visualizzate.

Una volta deciso quali informazioni mostrare e in che posizione dell'interfaccia mostrale, nel caso in cui queste non siano visibili, si è proceduto con lo studio delle possibili cause di questa mancata visualizzazione e su come modificare il documento XML di input attraverso l'applicazione delle regole di trasformazione del foglio di stile XSLT per poterle presentare tramite web.
Sono state definite conseguentemente le regole di trasformazione implementando le istruzioni dei template più adeguate da applicare all'interno del foglio di stile XSLT. Una volta implementato il foglio di stile con tutte le regole di trasformazione

necessarie, questo viene elaborato dal processore XSLT Saxon che ne esegue e ne applica le istruzioni generando il file di output XML desiderato compatibile con la struttura del documento elaborato da EVT.

L'elaborazione è stata eseguita dapprima per tutti i documenti che compongono il *corpus* epistolare belliniano ed infine da tutti questi documenti generati sono state integrate, tramite la procedura XInclude, le porzioni di codice XML di interesse per ogni lettera in un unico documento finale letto dal software EVT.

Il risultato finale permette la visualizzazione di tutte le informazioni desiderate sul software e una fruizione dell'edizione digitale ottimale.

6. Ristrutturazione del *corpus*

L'intero *corpus* belliniano, codificato secondo il modello di progetto, è stato rivisitato e sottoposto a un'ulteriore fase di adattamento al fine di ottimizzarne l'esperienza utente nell'ambiente web attraverso l'applicazione di specifiche regole aggiunte al foglio di stile XSLT.

Nello specifico, sono stati ristrutturati alcuni frammenti XML riguardanti:

- la registrazione delle coordinate per l'allineamento testo-immagine;
- la creazione dei blocchi di testo da riportare nel pop-up che appare cliccando sull'hotspot;
- la disposizione dei metadati codicologici e di dati relativi alla corrispondenza nonché la struttura dei metadati del *corpus* nella sua interezza (`msDesc`, `front`);
- le liste di entità nominate, di dati notevoli, le note e la bibliografia;
- l'eventuale presenza di pagine che non contengono testo autografo di Bellini.

Le trasformazioni precedentemente applicate non erano sufficienti per la fruizione di tutte le informazioni di interesse dell'edizione digitale. Quindi sono state rivisitate e modificate le istruzioni dei template dei fogli di stile XSLT.

EVT presenta numerose funzioni da poter sfruttare per la fruizione dell'edizione digitale. Grazie ai pulsanti dell'interfaccia web posizionati in alto è possibile accedere a due sezioni: la sezione *MSDesc* e la sezione *Info* (v. figura 5).

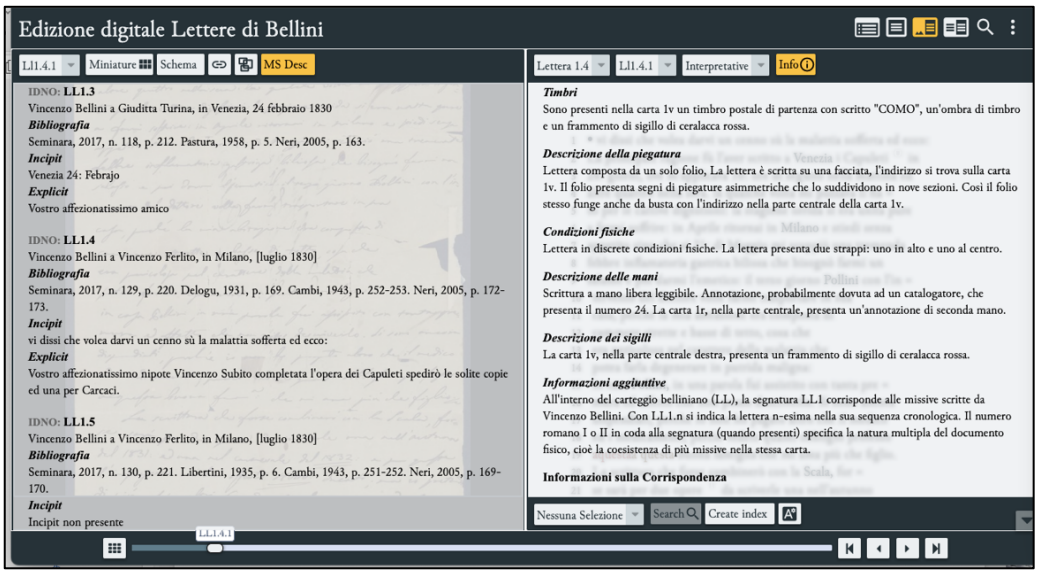

Figura 5. Sezioni *Info* e *MSDesc* su EVT.

Il gruppo di ricerca ha deciso di visualizzare nelle due sezioni alcune informazioni generali rispettivamente per l'intero *corpus* e per ogni lettera. In particolare la sezione *Info* riporta, per ogni lettera, le informazioni riguardanti il titolo, la lingua, la collocazione, il tipo di supporto usato, le sue condizioni fisiche, la presenza di timbri, di filigrana, di sigilli, di piegature, di annotazioni di altre mani e le informazioni sulla corrispondenza (mittente, destinatario, luogo di invio, eventuale luogo di ricezione e datazione). Mentre, la sezione *MS Desc*, relativa alla visualizzazione dei dati del manoscritto, riporta le informazioni riguardanti il titolo, la bibliografia di riferimento, l'incipit e l'explicit di ogni lettera.

Ciò che viene visualizzato in queste due sezioni si ottiene attraverso l'applicazione di alcune regole di trasformazione inserite nel foglio di stile XSLT. In dettaglio, si ottiene la ridisposizione degli elementi applicando alcune regole di trasformazione che ricodificano gli opportuni dati dell'intestazione (elemento `teiHeader`) da ogni singolo documento di input al contenuto del blocco `text/front` del documento di output. Per quanto riguarda le informazioni mostrate nella sezione *MS Desc,* vengono estratti i dati necessari dal documento XML ed inseriti in output sfruttando il content model dell'elemento `msContent`, in cui è stato possibile ricollocare la descrizione di ogni unità testuale nel documento di output mediante l'elemento `msItem`. Per quanto riguarda invece le informazioni mostrate nella sezione *Info*, vengono estratti i dati necessari dal documento XML ed inseriti all'interno dell'elemento `front` e quindi nel frontespizio di ogni lettera codificata.

L'edizione, inoltre, è ricca di entità nominate e fenomeni notevoli che vengono presentate in EVT in varie modalità grafiche (liste, sezioni in overlay, pop-up, ecc.). Ad esempio, per accogliere pienamente le informazioni relative alle note, alle opere citate e alla terminologia, ulteriori regole XSLT favoriscono la visualizzazione di aree in modalità pop-up. Per esempio, grazie all'inserimento di un apposito template per l'elemento `rs` con attributo `@type` e valore *work* (v. listato riportato nella sezione 13.1 dell'appendice), per ogni opera citata nel testo, cliccando accanto ad essa si apre un pop-up nel quale si possono visualizzare le informazioni riguardanti il titolo, il compositore, il librettista, la prima rappresentazione e le note che spiegano e danno approfondimenti sull'opera stessa (v. figura 6).

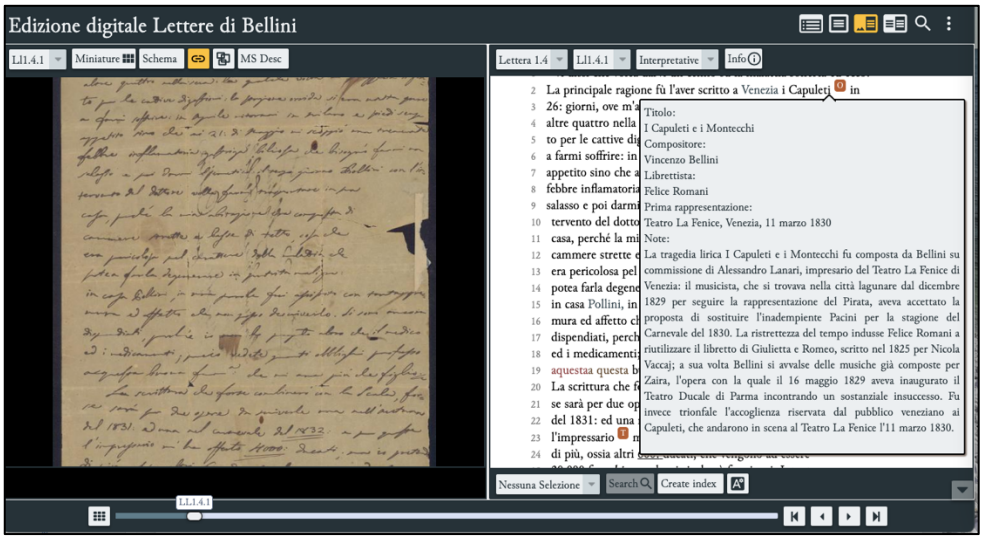

Figura 6. Visualizzazione su EVT del pop-up dell'opera *I Capuleti e i Montecchi* presente nella seconda riga della lettera LL1.4.

Infine, sono state condotte anche varie modifiche al codice del software EVT che hanno permesso la personalizzazione della resa grafica del pop-up soprattutto per le note di approfondimento terminologico e bibliografico. Queste note sono molto corpose tanto da dover cambiare dinamicamente la dimensione dei pop-up in modo che il browser attivi, in automatico ove necessario, le barre di scorrimento laterali.[60]

[60] Queste attività sono state svolte con altri membri del gruppo di lavoro, tra cui il dott. Pietro Sichera (CNR-ISTC di Catania).

7. Collegamento testo-immagine su EVT

7.1. Rappresentazione delle regioni di interesse facsimilari nel *corpus* belliniano

La codifica delle immagini del manoscritto avviene tramite gli elementi `facsimile`[61] e `surface`[62] in cui ogni riga del testo, che corrisponde ad un'area dell'immagine, è codificata invece con l'elemento `zone`.[63]

All'interno dell'elemento `zone` sono presenti gli attributi che registrano i valori dei punti o delle coordinate cartesiane che ne definiscono l'area che le delimita: ovvero `@points` per i punti e `@ulx` e `@uly` per le coordinate x e y dell'angolo superiore sinistro e `@lrx` e `@lry` per le coordinate x e y dell'angolo inferiore destro (ottenendo la diagonale di un rettangolo).

Ogni zona inoltre può essere associata sia alle righe del testo scritto da Vincenzo Bellini sia agli hotspot. Gli hotspot vengono usati per inserire annotazioni testuali riguardanti l'elemento delineato nell'immagine e possono far riferimento a timbri, segni sul supporto, annotazioni e testi di altre mani, ecc.

Per definire il tipo di zona che si codifica nel documento XML all'interno dell'elemento `zone` si utilizza l'attributo `@rendition` il cui valore può essere uguale a *Line*, nel caso si tratti di una riga di testo, oppure a *Hotspot,* nel caso in cui si tratti di un hotspot.

Per collegare il testo della trascrizione alle aree corrispondenti dell'immagine viene assegnato un identificatore univoco a ciascun elemento del `facsimile` usando l'attributo `@xml:id` e l'attributo `@start`.

La codifica dell'elemento del testo a cui si collega questa porzione di immagine presenta invece l'attributo `@facs` che ha come valore quello dell'identificatore a cui fa riferimento.

---

[61] L'elemento `facsimile` definisce una superficie scritta di un'immagine e costituisce una rappresentazione digitale di una fonte con testo scritto espressa attraverso una serie di immagini, piuttosto che attraverso la trascrizione e la codifica del testo.

[62] L'elemento `surface` definisce una superficie di scrittura in termini di uno spazio espresso attraverso un sistema di coordinate. Opzionalmente può essere inserita una collezione di elementi `graphic` che rappresentano graficamente lo spazio indicato. È possibile caratterizzare la superficie con successivi elementi `zone` che rappresentano specifiche regioni di interesse su di essa.

[63] L'elemento `zone` definisce qualsiasi area bidimensionale all'interno di un elemento `surface`.

7.2. Resa grafica degli hotspot e delle righe del testo

Il software EVT permette di mostrare le aree dell'immagine che corrispondono sia agli hotspot sia alle righe del testo di Bellini. Questa funzione si può attivare cliccando uno dei due pulsanti posizionati nella barra dei comandi nella parte superiore sotto al titolo dell'edizione digitale.

Cliccando sul pulsante rappresentato dal disegno di una graffetta vengono mostrate delle sezioni che corrispondono alle righe del testo autografo di Bellini. Invece cliccando sull'altro pulsante, in cui sono rappresentati dei quadratini, vengono mostrati dei riquadri rossi in trasparenza che delineano le aree degli hotspot (v. figura 7).

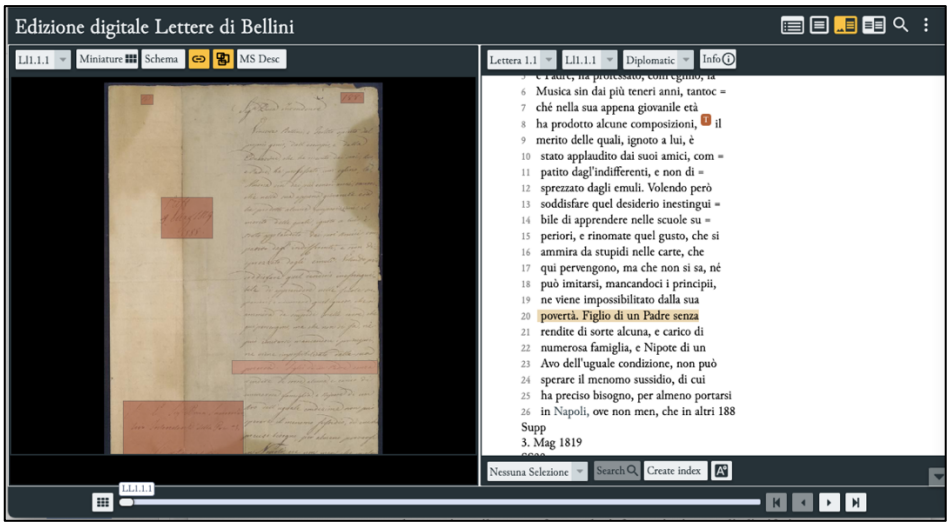

Figura 7. Visualizzazione degli hotspot e delle righe della lettera LL1.1 su EVT.

7.3. Resa grafica dei pop-up degli hotspot

Per poter visualizzare il contenuto testuale associato ai vari hotspot presenti nel *corpus* belliniano viene sfruttata una funzione di EVT che consente di mostrare sia le aree dell'immagine di ogni hotspot sia il paragrafo che contiene il testo associato ad esso attraverso un pop-up. Attivando la funzione *hotspot* dal pulsante corrispondente, ogni area di hotspot appare delineata in un riquadro rosso in trasparenza. Cliccando su quest'area si apre un pop-up che contiene il paragrafo con le informazioni testuali relative all'hotspot (v. figura 8).

Figura 8. Visualizzazione del testo che descrive l'hotspot evidenziato nella lettera LL1.16 su EVT.

Il contenuto del pop-up è ripreso dal blocco `div`, contenuto nell'elemento `back`, con attributo `@type` uguale a *'hotspot'*. Dentro questo blocco saranno contenuti tanti blocchi `div`, uno per ogni hotspot con al suo interno il paragrafo corrispondente.

Per ottenere l'inserimento di questa porzione di codice nel documento XML elaborato da EVT si effettua una nuova trasformazione del documento XML di input attraverso l'aggiunta di una nuova regola di trasformazione XSLT nel foglio di stile.

Il template creato consente di ricostruire l'elemento `back` (che va a sostituire quello presente nel documento XML di input) nel quale inserire solo la codifica delle informazioni da mostrare (v. listato nella sezione 13.1 dell'appendice).

All'interno dell'elemento `back` viene inserito un elemento di blocco `div` con l'attributo `@type` con valore uguale a *hotspot.* Come specificato nel listato XSLT seguente, selezionando solo gli elementi `zone` che hanno l'attributo `@type` con valore *Hotspot* si assegna a una variabile, che si chiama `$zone2`, il valore dell'attributo `@xml:id` di tali zone e si crea un altro blocco `div`.

Quest'ultimo blocco `div` ha a sua volta un attributo `@xml:id` che ha come valore quello della variabile `$zone2` concatenata alla lettera *h* finale e un attributo `@facs` che ha come valore sempre quello di `$zone2` preceduto da un cancelletto in modo da definire il collegamento per quella zona. In questo blocco il processore va poi a copiare, all'interno di un paragrafo, il testo contenuto negli elementi del `teiHeader` solo se hanno il valore dell'attributo `@facs` (il cui valore è associato alla variabile

`$facs`) uguale al valore della parte di stringa che segue il carattere '#' dell'attributo `@xml:id` della zona a cui fa riferimento. Ogni blocco è associato alla porzione di testo corrispondente tramite l'attributo `@facs`. Di seguito viene riportato il codice descritto sopra.

```
[…]
<xsl:template match="tei:back">
<xsl:element name="back" namespace="http://www.tei-c.org/ns/1.0">
<xsl:element name="div" namespace="http://www.tei-c.org/ns/1.0">
<xsl:attribute name="type">hotspot</xsl:attribute>
<xsl:variable name="facs" select="//tei:teiHeader//*[@facs]"/>
<xsl:variable name="zone" select="//tei:zone[@rendition='HotSpot']"/>
<xsl:for-each select="$zone">
   <xsl:variable name="zone2" select="@xml:id"/>
   <xsl:variable name="newid" select="concat ($zone2, 'h')"/>
   <xsl:variable name="newfacs" select="concat('#', $zone2)"/>
   <xsl:element name="div" namespace="http://www.tei-c.org/ns/1.0">
      <xsl:attribute name="xml:id">
           <xsl:value-of select="$newid"/>
      </xsl:attribute>
      <xsl:attribute name="facs">
           <xsl:value-of select="$newfacs"/>
      </xsl:attribute>
      <xsl:for-each select="$facs">
           <xsl:variable name="facs3" select="@facs"/>
           <xsl:variable name="facs4" select="translate($facs3, '#',
           '')"/>
           <xsl:if test="$facs4=$zone2">
                <xsl:element name="p" namespace="http://www.tei-
           c.org/ns/1.0">
                     <xsl:value-of>
                          <xsl:apply-templates mode="hotspot"/>
                     </xsl:value-of></xsl:element>
           </xsl:if>
      </xsl:for-each></xsl:element>
</xsl:for-each> […]
```

Per evitare ripetizioni di testo nei pop-up degli hotspot, come mostrato nel codice XSLT che segue, la variabile `$facs` seleziona tutti gli elementi che hanno un attributo `@facs` presenti solo nell'elemento `teiHeader`, evitando così di recuperare anche quelli in `body` e in `front`.

```
<xsl:variable name="facs" select="//tei:teiHeader//*[@facs]"/>
```

7.4. Normalizzazione delle coordinate delle zone

Vista l'arbitrarietà delle dimensioni delle immagini e delle carte nel *corpus* belliniano, per ottenere una corretta funzione di allineamento testo-immagine in EVT, è stata indispensabile un'attività di normalizzazione delle coordinate cartesiane registrate nelle zone di interesse. Tale normalizzazione è stata eseguita mediante lo sviluppo di una regola contenuta nel foglio di stile XSLT.

La regola è applicata all'elemento `surface` (v. listato nella sezione 13.1 dell'appendice) e la normalizzazione è stata calcolata in funzione del rapporto tra la dimensione di ogni singola immagine (che è indicata come valore dell'attributo `@width` dell'elemento `graphic`) e un coefficiente costante; successivamente il valore di ogni coordinata (i cui valori sono indicati negli attributi `@ulx`, `@uly`, `@lrx`, `@lry` degli elementi `zone`) si divide per il rapporto precedentemente calcolato.

Il valore del coefficiente di calcolo è riportato nel file di configurazione di EVT, nel parametro *imageNormalizationCoefficient* nelle opzioni *imageViewerOptions*. Nel nostro caso il valore è 723 ed è stato stabilito prendendo il valore dell'attributo `@width` dell'elemento `graphic` più basso tra tutte le lettere del *corpus*.

I valori delle nuove coordinate saranno così aggiornati in modo da ottenere una corretta funzione di allineamento delle righe del testo manoscritto belliniano con l'immagine su EVT.

Lo sviluppo della regola di trasformazione avviene in questo modo: prima di tutto si imposta il template per l'elemento `surface`, poi vengono create le variabili a cui sono assegnati i valori degli elementi e degli attributi necessari per la creazione del nuovo elemento `surface` (`@idno`, `@n`, `@xml:id`, `pb`).

```
<xsl:variable name="idno"
select="//tei:idno[@type='inventory']"/>
<xsl:variable name="n" select="@n"/>
<xsl:variable name="newN" select="concat ($idno, '.', $n)"/>
<xsl:variable name="id" select="@xml:id"/>
<xsl:variable name="pb" select="//tei:body//tei:pb"/>
```

Viene poi creato l'elemento `surface` ed i suoi attributi (`@n`, `@xml:id`, `@corresp`).

```
<xsl:element name="surface" namespace="http://www.tei-
c.org/ns/1.0">
<xsl:attribute name="n">
  <xsl:copy-of select=" $newN"/>
</xsl:attribute>
<xsl:attribute name="xml:id">
  <xsl:copy-of select="$id"/>
</xsl:attribute>
<xsl:attribute name="corresp">
  <xsl:for-each select="$pb">
    <xsl:if test="@n=$n">
      <xsl:variable name="facs" select="@xml:id"/>
      <xsl:variable name="newfacs" select="concat('#', $facs)"/>
      <xsl:value-of select="$newfacs"/>
    </xsl:if> </xsl:for-each></xsl:attribute>
```

Dentro questo elemento viene applicata la regola sviluppata per la creazione dell'elemento `graphic`, che rappresenta la codifica della superficie di ogni pagina della lettera.

```
<xsl:apply-templates select="tei:graphic"/>
```

Successivamente vengono impostati i valori delle variabili necessarie per creare gli elementi `zone` all'interno di `surface`.

```
<xsl:variable name="width" select="tei:graphic/@width"/>
<xsl:variable name="widthnew" select="substring-before($width,
'px')"/>
<xsl:variable name="coeff" select="number(723)"/>
<xsl:variable name="rapp" select="number($widthnew) div
$coeff"/>
```

Quindi il valore dell'attributo `@width` che rappresenta il valore della dimensione, il valore del coefficiente fisso che è 723 e il rapporto ottenuto dividendo il valore dell'attributo `@width` e questo coefficiente che serviranno per calcolare il valore aggiornato delle coordinate delle zone.
A questo punto viene eseguita l'istruzione `xsl:for-each` per ogni elemento `zone` per ricostruire tutte le zone con le nuove coordinate utili ad ottenere l'allineamento riga-testo su EVT.
Dapprima si impostano le variabili da utilizzare con i valori degli attributi delle zone (`@xml:id`, `@corresp`, `@rendition`, `@ulx`, `@uly`, `@lrx`, `@lry`).

```
<xsl:for-each select="tei:zone ">
<xsl:variable name="id" select="@xml:id"/>
<xsl:variable name="corresp" select="concat ('#', $id, 'h')"/>
<xsl:variable name="rendition" select="@rendition"/>
<xsl:variable name="ulx" select="@ulx"/>
<xsl:variable name="uly" select="@uly"/>
<xsl:variable name="lrx" select="@lrx"/>
<xsl:variable name="lry" select="@lry"/>
```

Poi viene creato l'elemento `zone`, gli attributi `@ulx`, `@uly`, `@lrx`, `@lry` calcolando il valore di ogni coordinata e gli attributi `@xml:id`, `@rendition`, `@corresp` e `@start`.
Per calcolare il nuovo valore di ogni coordinata si divide il valore che aveva inizialmente ogni coordinata per il rapporto che è stato calcolato dividendo il valore della dimensione specificata nell'attributo `@width` col coefficiente fisso che è impostato a 723.

```
<xsl:element name="zone" namespace="http://www.tei-
c.org/ns/1.0">
  <xsl:attribute name="ulx">
    <xsl:variable name="result" select="$ulx div $rapp"/>
    <xsl:value-of select="format-number($result, '#.00')"/>
  </xsl:attribute>
  <xsl:attribute name="uly">
  <xsl:variable name="result" select="$uly div $rapp"/>
    <xsl:value-of select="format-number($result, '#.00')"/>
  </xsl:attribute>

[…]

  <xsl:if test="$rendition='HotSpot'">
    <xsl:attribute name="corresp">
      <xsl:value-of select="$corresp"/>
    </xsl:attribute>
    <xsl:attribute name="start">
      <xsl:value-of select="$corresp"/>
    </xsl:attribute>
  </xsl:if>
</xsl:element>
```

In seguito, è stato necessario gestire anche la presenza di un annidamento di zone nella codifica dei documenti di input.[64] Questo è dovuto alla codifica delle missive con l'elemento `surface`, che contiene la codifica dell'immagine del manoscritto divisa in due zone: una che contiene il primo testo scritto in verticale e l'altra che contiene il secondo testo scritto in orizzontale (v. figura 9).
L'elemento `zone` padre codifica ogni testo e l'elemento `zone` figlio codifica ogni riga del testo. Ogni zona, quindi, è a sua volta composta dagli elementi `zone` che codificano le rispettive righe dell'immagine associate al corrispondente testo codificato nell'elemento `body`.

[64] Le lettere in questione sono la LL1.14, la LL1.34 e la LL1.13.II.

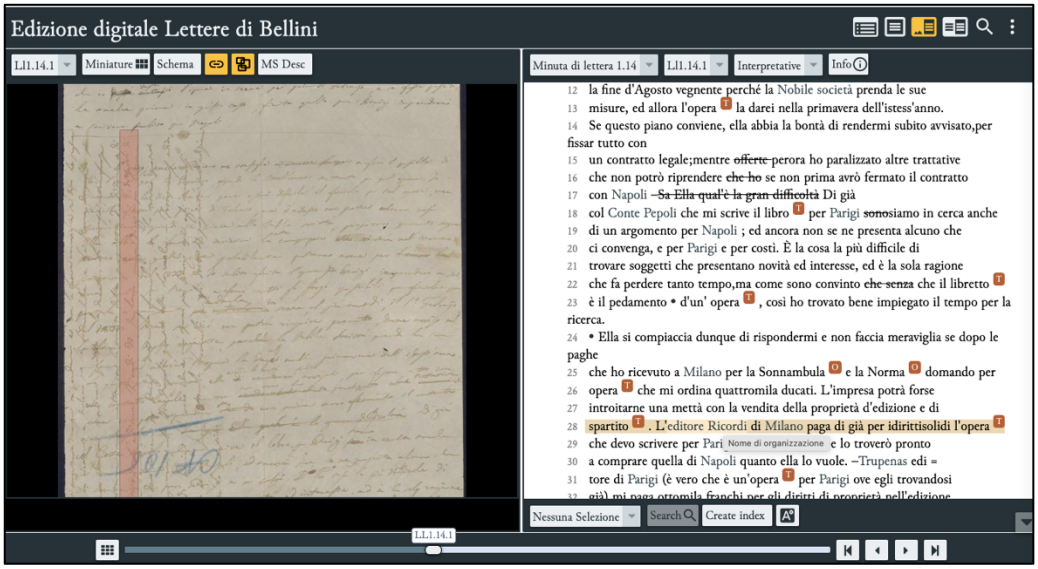

Figura 9. Visualizzazione della lettera LL1.14 su EVT.

Così per calcolare i nuovi valori degli attributi `@ulx`, `@uly`, `@lrx`, `@lry` di ogni elemento `zone` che codifica la riga si aggiunge un'istruzione al template.

```
<xsl:if test="exists(tei:zone/tei:zone)">
    <xsl:for-each select="tei:zone/tei:zone"> [...]
```

Questa istruzione viene eseguita solo se sono presenti elementi zone figli annidati e per ognuno di questi ricreano gli elementi `zone` con i corrispondenti attributi ricalcolando anche per queste zone il valore delle coordinate per ottenere l'allineamento desiderato su EVT.

7.5. Altre trasformazioni effettuate

Accanto alla definizione delle regole di trasformazione descritte nei capitoli precedenti, per poter fruire di altre informazioni riguardanti l'edizione digitale sono state aggiunti altri templates al foglio di stile.

Grazie ad EVT è possibile visualizzare la bibliografia completa selezionando tramite l'apposito pulsante laterale dapprima la voce '*Informazioni sul progetto'* ed in seguito la voce *'Bibliografia'* (v. figura 10).

Figura 10. Visualizzazione della bibliografia su EVT.

Al fine di conservare e visualizzare le informazioni bibliografiche nell'ambiente Web è stata aggiunta una nuova regola XSLT al foglio di stile (v. figura 11 e listato nella sezione 13.1 dell'appendice) che consente di copiarla dal file *TEI-ListBibl.xml* ed aggiungerla al documento elaborato da EVT creando un ulteriore frammento XML con un elemento `div` e l'attributo `@type` con valore uguale a *biblCompleta* all'interno dell'elemento `back` dell'ultima lettera.

```
<xsl:if test="//tei:idno[@type='inventory']='LL1.35'">
    <xsl:element name="div" namespace="http://www.tei-c.org/ns/1.0">
        <xsl:attribute name="type">biblCompleta</xsl:attribute>
        <xsl:element name="listBibl" namespace="http://www.tei-c.org/ns/1.0">
            <xsl:variable name="listbibl" select="doc('lists/TEI-ListBibl.xml')//tei:back//tei:div//tei:listBibl"/>
            <xsl:copy-of select="$listbibl"/>
        </xsl:element>
    </xsl:element>
</xsl:if>
```

Figura 11. Porzione di codice presente nel template del foglio di stile XSLT che consente di visualizzare la bibliografia su EVT.

Per le pagine che non contengono testo autografo di Bellini è stato previsto un altro template nel foglio di stile XSLT che prevede l'inserimento di un blocco `div` contenente un paragrafo con valore testuale *"La facciata non presenta testo autografo di Bellini relativo a questa lettera"* (v. listato nella sezione 13.1 dell'appendice).

Allo scopo è stata definita una regola di trasformazione inserita nel foglio di stile XSLT per l'elemento `pb` (v. figura 12). Il blocco di testo viene inserito dal template solo nel caso in cui la lettera non presenti testo autografo di Bellini.

Figura 12. Porzione di codice del template del foglio di stile XSLT che consente di visualizzare la scritta indicata nelle pagine dell'edizione in cui non compare testo autografo di Bellini.

Per verificare la condizione il template controlla ogni elemento `surface` collegato al testo tramite l'attributo `@facs` dell'elemento `pb`. Per ognuno di questi elementi verifica, poi, se contiene elementi `zone` che presentano l'attributo `@rendition` con valore uguale a *Line.* Se tale valore è presente la pagina della lettera ha testo autografo di Bellini e quindi il paragrafo non viene inserito, se invece non è presente il foglio considerato non contiene testo autografo di Bellini e quindi inserisce il blocco `div` con all'interno il paragrafo con la scritta inerente. In questo caso sono state inserite due condizioni, una per le lettere che presentano annidamenti di zone nell'elemento `surface` e un'altra per le restanti (v. figura 12).

8. Tecnologie XML-based per la creazione di viste web per l'edizione digitale

Il lavoro svolto per l'ottimizzazione della fruizione del *corpus* belliniano sul web attraverso il software EVT ha prodotto un ottimo risultato. Tuttavia, sono state incluse numerose operazioni che hanno comportato tempi di lavorazione piuttosto lunghi per ottenere quanto desiderato. In aggiunta alle soluzioni descritte nei capitoli precedenti, ci si è chiesto se fosse possibile avvalersi dell'impiego di ulteriori tecnologie per la pubblicazione dell'edizione digitale sul web.

Tra gli obiettivi del progetto di tesi, infatti, si annovera anche quello di produrre ulteriori prototipi di fruizione del *corpus* mediante diversi approcci all'elaborazione dei documenti XML. La sperimentazione si è concentrata quindi sull'adozione di tecnologie XML-based che consentano una maggiore flessibilità rispetto all'utilizzo

di EVT. Come ampiamente descritto, la scelta principale per la visualizzazione e la fruizione delle lettere di Bellini è ricaduta sul software EVT. Ma è possibile adottare anche altre tecnologie?

La soluzione più immediata è quella di realizzare con fogli di stile XSLT un ambiente web per la trasformazione e la visualizzazione. Fino a un po' di tempo fa tale soluzione sarebbe stata poco efficiente da realizzare, ma attualmente il processo può essere semplificato dall'utilizzo di una libreria Javascript, Saxon-JS2, disponibile sia per l'esecuzione client-side da browser sia server-side per Node.js.

Altre tecnologie XML-based come XQuery ed eXist-db rendono estremamente flessibile ed efficace l'elaborazione, la pubblicazione e la visualizzazione dei dati codificati per mezzo del vocabolario XML-TEI.

8.1. Node.js e NPM

Node.js è un *framework*[65] open source sviluppato per l'esecuzione di codice Javascript in applicazioni web server-side. Mentre in origine Javascript era usato principalmente lato client,[66] in questo scenario invece i programmi scritti in Javascript (in genere incorporati in una pagina HTML) vengono interpretati sia dal client (tipicamente un programma incorporato nel browser) sia dal server.

Infatti, Node.js consente di usare Javascript anche per scrivere codice da eseguire lato server[67] prima che la pagina venga inviata al browser dell'utente. Tale approccio consente così di sviluppare applicazioni web adottando un unico linguaggio di programmazione con un'architettura orientata agli eventi che rende possibile una efficiente comunicazione asincrona tra le varie componenti del programma. Con il comando `node` si può eseguire un'applicazione contenuta in un file indicato come parametro.

---

[65] Un *framework* è un software che fornisce i servizi necessari per eseguire un programma.

[66] Indica operazioni di elaborazioni eseguite da un client, ovvero da un componente software presente in una macchina *host,* che accede ai servizi e alle risorse di un'altra componente detta *server* usando protocolli di comunicazione. In un'architettura *client-server* un terminale si connette a un server per la fruizione di un certo servizio con altri client. La presenza di un server permette a un certo numero di client di condividere le risorse lasciando che sia il server a gestire gli accessi alle risorse.

[67] Operazione compiuta da un server. In genere il server è un programma o software che gira su una macchina rimanendo in ascolto su determinate porte e raggiungibile da un computer client.

Node.js permette l'utilizzo di un ulteriore comando, npm, acronimo di Node Package Manager. Npm è un gestore di pacchetti per l'ambiente Node.js e i moduli sviluppati in Javascript. Esso consiste in un repository di librerie scritte per essere usate con Node.js, chiamato *npm registry*, utile per scaricare ed installare numerose applicazioni e plugin tramite la riga di comando `npm install`. Npm è installato con Node.js e tutti i pacchetti sono descritti da un documento in formato JSON.

Dopo aver installato Node.js si può configurare l'applicazione che si vuole creare, eseguendo da terminale la riga di comando `npm init` e compilando tutti i dati richiesti per la creazione dell'applicazione (tra cui package name, version, description, entry point, text comand, git repository, keywords, license), generando così il file package.json che ci serve per l'applicazione.

8.2. Saxon-JS 2

Saxon-JS è un processore XSLT 3.0 ad alte prestazioni scritto in Javascript e conforme alle più recenti specifiche W3C. Saxon-JS2, che è l'ultima versione, è disponibile per due ambienti Javascript: il browser (lato client) e Node.js (lato server) e viene inoltre fornito di un compilatore XSLT che funziona in maniera completamente indipendente dal processore Saxon a nativo sviluppato in Java (e usato per eseguire le trasformate XSLT descritte nei capitoli precedenti). Grazie a questa tecnologia è possibile costruire un'applicazione web elaborando un documento XML direttamente nel browser seguendo le regole indicate nel foglio di stile che saranno eseguite direttamente al caricamento della pagina web. Il browser esegue lato client il processore Saxon-JS2, che elabora a sua volta un documento XML, generando la pagina web ottenuta applicando le regole definite nel foglio di stile XSLT compilato nel formato SEF.

Il formato SEF è un formato di esportazione in formato JSON e non XML riconosciuto da Saxon-JS. Nella prima versione di Saxon-JS il formato SEF veniva rappresentato in XML, in Saxon-JS 2 invece viene rappresentato nel formato JSON che è più compatto e adatto ad essere elaborato da programmi scritti in Javascript.

Il file di esportazione del foglio di stile XSLT (file SEF) si genera invece lato server utilizzando un compilatore javascript XSL-to-SEF che esegue il foglio di stile XSLT e lo esporta nel formato SEF.

In figura 13 viene mostrato il processo eseguito da Saxon-JS.

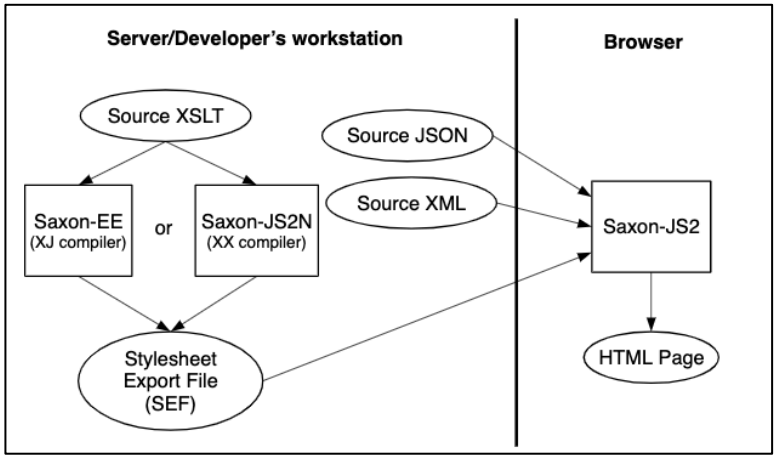

Figura 13. Diagramma che mostra l'elaborazione e la compilazione da parte del server e l'esecuzione del browser usando Saxon-JS.

8.3. Installazione ed esecuzione di Saxon-JS 2 su Node.js

Grazie a Node.js è possibile installare tramite NPM due pacchetti utili a realizzare l'applicazione web desiderata: Saxon-JS e XSLT 3.0 eseguendo dal terminale le due righe di comando `npm install saxon-js` e `npm install xslt3`.

Il pacchetto Saxon-JS (*SaxonJS2N.js*) include tutto il codice necessario per compilare ed eseguire fogli di stile XSLT 3.0 nell'ambiente Node.js, inclusa un'API che consente alle applicazioni di invocare la compilazione e l'esecuzione di fogli di stile. Il pacchetto XSLT3 (*xslt3.js*) fornisce un'interfaccia a riga di comando che consente di compilare ed eseguire trasformazioni XSLT 3.0 da terminale.[68]

Una volta effettuate le installazioni si può generare il file di esportazione SEF usando un compilatore javascript XSL-to-SEF che prende come input il file XSLT sorgente e genera come output il file SEF.

Saxonica offre due compilatori per generare file SEF: il compilatore XJ, fornito come parte del prodotto Saxon-EE, che è scritto in Java e il compilatore XX, fornito invece con Saxon-JS 2, che è scritto in XSLT ed effettua chiamate su un parser XPath scritto in Javascript. I file SEF che generano sono codificati in JSON e il vocabolario SEF è lo stesso per entrambi i compilatori, tuttavia l'output non è identico. Il compilatore XX è quindi incluso nella libreria di Saxon-js che è stato installato e per generare il foglio di esportazione SEF a questo punto si deve eseguire dal terminale la seguente riga di comando:

[68] <https://saxonica.com/saxon-js/documentation/index.html#!about/components>.

```
xslt3 -t -xsl:stylesheet.xsl -export:stylesheet.sef.json
-nogo
```
[69]

8.4. Installazione ed esecuzione di Saxon-JS 2 sul browser

Saxon-JS, per l'esecuzione sul browser, è distribuito sotto forma di codice Javascript compresso per essere recuperato ed eseguito dal browser per interagire direttamente con le pagine HTML. La libreria si può scaricare dalla pagina web Saxonica[70] ed include una versione completa (*SaxonJS2.js*) e una ridotta (*SaxonJS2.rt.js*).

Per l'esecuzione si deve inserire nell'elemento `head` del file HTML lo script che include la libreria per la versione completa o ridotta. Per quella completa si riporta di seguito un esempio:

```
<script type="text/javascript" language="javascript"
src="SaxonJS2.js"></script>
```

Per eseguire il file di esportazione SEF, invece, si deve inserire lo script Javascript (v. figura 14) che al caricamento della pagina applica il metodo `trasform`.

Questo metodo fornisce al processore le opzioni di trasformazione e quindi il file SEF da compilare, il documento di input da trasformare e la modalità sincrona o asincrona della trasformazione.[71]

```
<script type="text/javascript" language="javascript" src="SaxonJS2.js"></script>
<script>
    window.onload = function() {
        const sourcexml=["lettere/LL1_1.xml","lettere/LL1_2.xml"];
        SaxonJS.transform({
        stylesheetLocation: "trasforma.sef.json",
        sourceLocation: sourcexml[0]
        }, "async");
    }
</script>
```

Figura 14. Script Javascript che consente di eseguire il file di esportazione SEF al caricamento della pagina web.

[69] L'istruzione *-t* fa in modo che il comando visualizzi i messaggi di avanzamento, l'istruzione *-xsl:stylesheet.xsl* identifica il foglio di stile di input, l'istruzione *-export:stylesheet.sef.json* identifica il file in cui verrà scritto il file SEF generato e l'istruzione *-nogo* sopprime qualsiasi tentativo di eseguire il foglio di stile.

[70] <https://www.saxonica.com/download/javascript.xml>.

[71] *`StylesheetLocation`* fa riferimento al file SEF e *`sourceLocation`* al documento XML da elaborare a cui applicare il foglio di stile.

8.5. Implementazione dell'applicazione *Bellini Digital Correspondence* attraverso l'utilizzo di Saxon-js

Per realizzare l'applicazione *Bellini Digital Correspondence* è stata utilizzata la tecnologia XSLT e sperimentata la libreria Saxon-JS, secondo il processo descritto nella figura 15.

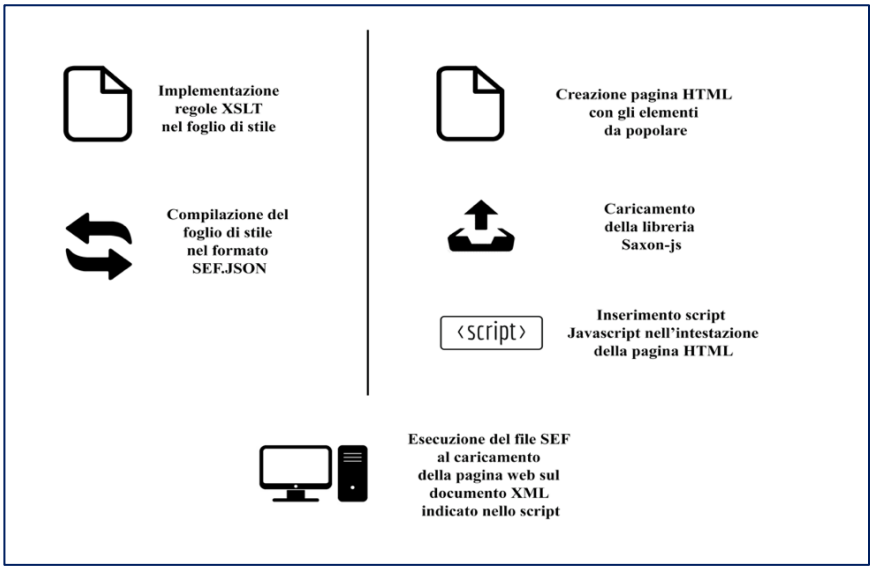

Figura 15. Processo di creazione dell'applicazione con la tecnologia Saxon-JS.

Innanzitutto, è stata creata l'applicazione con Node.js eseguendo dal terminale il comando `npm init`. In seguito, sono stati installati gli strumenti XSLT3 e Saxon-JS ed è stato implementato il foglio di stile XSLT contenente le regole di trasformazione da applicare. Successivamente il documento XSLT è stato esportato in SEF mediante il comando:

```
xslt3 -t -xsl:bellini.xsl -export:trasforma.sef.json -nogo.
```

Una volta caricata la libreria Saxon-JS2 nel corpo della pagina HTML è stato inserito anche lo script Javascript che esegue il file SEF al caricamento della pagina (v. figura 14). Lo script contiene la funzione `SaxonJS.transform` che permette di invocare una trasformazione XSLT e di eseguirla.

La proprietà *stylesheetLocation* indica l'URI del foglio di stile precompilato SEF in formato JSON da eseguire, la proprietà *sourceLocation* indica l'URI che individua il documento XML di input, che nel caso del prototipo sviluppato per BDC è il documento *LL1_1.xml*, ovvero il primo elemento dell'array memorizzato nella

variabile *sourcexml* che rappresenta la lista delle lettere codificate in XML-TEI. Con il letterale *async* si attiva la funzione di trasformazione in modo asincrono.[72]

La pagina HTML ha una struttura di base che contiene alcuni elementi, identificati con l'attributo `@id`, inizialmente vuoti che saranno popolati sulla base delle istruzioni XSLT contenute nelle regole specificate nel foglio di stile indicate nell'istruzione `xsl:result-document` dove l'attributo `@href` indica `l'@id` del blocco in cui inserire il risultato dell'istruzione (vedere i listati *index.html* e *bibliografia.html* nella sezione 13.2 dell'appendice).

Ad esempio nel blocco `h1`, identificato con l'attributo `@id` con valore *title*, la regola inserisce il titolo della pagina. Nel blocco `div`, identificato con l'attributo `id` con valore *'text'*, per la pagina *lettera.html* la regola inserisce il titolo, le immagini e il testo della prima missiva e per la pagina *bibliografia.html* mostra l'elenco della bibliografia dell'intero *corpus* di lettere di *Bellini Digital Correspondence*.

Per il blocco `div`, identificato con l'attributo `@id` che ha valore *infosupporto*, la regola inserisce le informazioni aggiuntive riguardanti la prima lettera, tra cui le informazioni sul supporto.

Il foglio di stile XSLT da eseguire nel browser con Saxon-JS include dei template con le definizioni delle regole di trasformazione da applicare in cui sono indicate le istruzioni da eseguire (vedere i listati *bellini.xsl* e *bibliografia.xsl* nella sezione 13.2 dell'appendice).

L'istruzione `xsl:result-document` genera un albero di output dalla trasformazione XSLT e individua la porzione del DOM HTML dove l'albero generato deve essere aggiunto all'HTML della pagina sottoposta al rendering. Si possono così ottenere frammenti di HTML da inserire nello scheletro statico della pagina nei punti indicati all'interno del documento XSLT.

Il frammento di codice verrà posizionato nel punto desiderato attraverso l'attributo `@href` stante l'elemento in cui va inserita la porzione di codice. Questo elemento sarà poi popolato dal processore XSLT durante l'esecuzione del foglio di stile.[73]

I fogli di stile realizzati per Saxon-JS2 dichiarano un namespace con prefisso `ixsl` che definisce funzionalità aggiuntive per interagire con il DOM delle pagine HTML.

---

[72] <https://www.saxonica.com/saxon-js/documentation/index.html#!api/transform/execution>.

[73] <https://saxonica.com/saxon-js/documentation/index.html#!browser/result-documents>.

Per aggiungere un frammento di HTML si usa il metodo `ixsl:append-content` per sostituirlo invece si usa il metodo `ixsl:replace-content`.
Quando invece sono necessarie effettuare modifiche che interessano gli attributi o proprietà di stile si possono usare le istruzioni: `ixsl:set-attribute`, `ixsl:remove-attribute` e `ixsl:set-style`.
Il metodo `ixsl:set-attribute` imposta il nome e il valore di un attributo dell'elemento corrente o su un altro elemento specificato.
Il metodo `ixsl:remove-attribute` rimuove un attributo con un determinato nome dall'elemento corrente o da un altro nodo dell'elemento specificato.
Il metodo `ixsl:set-style` imposta una proprietà di stile di un elemento HTML.[74]

Nel foglio di stile usato per l'applicazione creata per *Bellini Digital Correspondence* è stata impiegata l'istruzione `xsl:result-document` e il metodo `ixsl:replace-content` che va a sostituire il codice HTML presente nell'elemento che ha come identificatore l'id `text` ed al suo interno inserisce un elemento `h2` che seleziona e riporta il titolo dal documento XML leggendo il valore dell'attributo `@tei:title`, dopodiché per ogni facciata della lettera, codificata con l'elemento `pb`, crea un blocco `div` che contiene l'immagine e il testo corrispondente (come riportato nel codice XSLT che segue).

[...][75]

```
<xsl:result-document href="#text" method="ixsl:replace-content">
 <h2>
  <xsl:value-of select="//tei:idno[@type='inventory']"/>-
  <xsl:value-of select="//tei:titleStmt//tei:title"/>
 </h2>
 <xsl:for-each select="//tei:body//tei:pb">
 <xsl:choose>
  <xsl:when test="@n='1'">
   <div id="pagina1" class="pagina">
    <div id="image1" class="text-left">
```

[74] <https://saxonica.com/saxon-js/documentation/index.html#!ixsl-extension/instructions>.

[75] Il documento completo può essere visualizzato nell'appendice o recuperato al seguente indirizzo web: <https://github.com/santapellino/Progetto-tesi-Santa-Pellino/blob/main/Applicazione%20BDC%20Saxon/bellini.xsl>.

```
        <xsl:variable name="link" select="concat('LL1-1_000',@n,
         '.jpg' )"/>
        <xsl:element name="img">
          <xsl:attribute name="src">
            <xsl:value-of select="$link"/>
          </xsl:attribute>
        </xsl:element>
      </div>
      <div id="text1" class="text-right">
        <b>Pagina 1</b><br/>
        <xsl:copy-of
         select="//tei:body/tei:div[@xml:id='info_dest']"/><br />
        <xsl:copy-of
         select="//tei:body/tei:div/tei:ab[@n='ab_02']"/>
      </div>
    </div>
</xsl:when>[...]
```

Nelle seguenti figure si possono visualizzare le interfacce dell'applicazione ottenute:

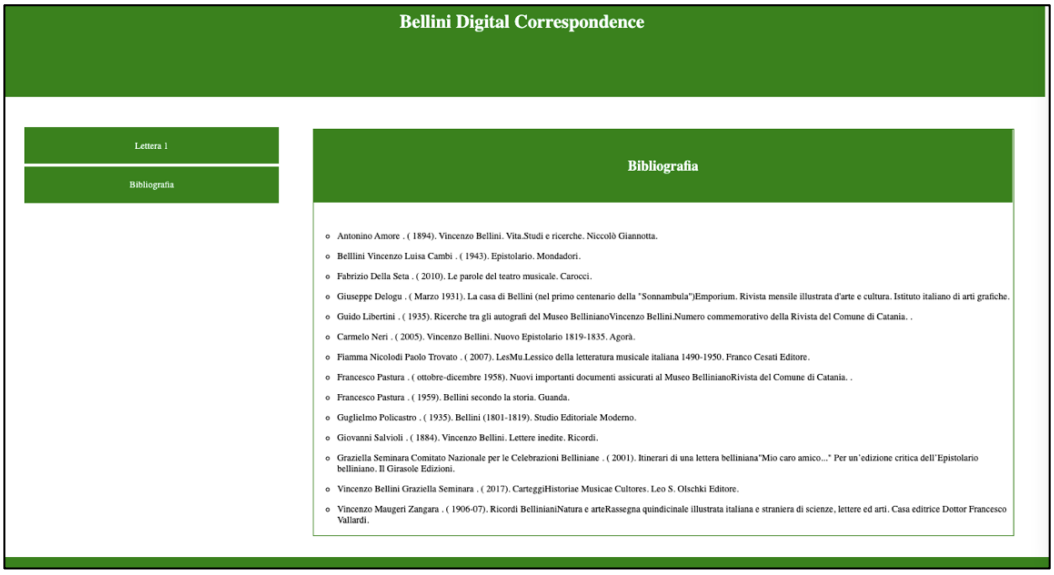

Figura 16. Applicazione realizzata usando Saxon-js che consente di visualizzare la bibliografia.

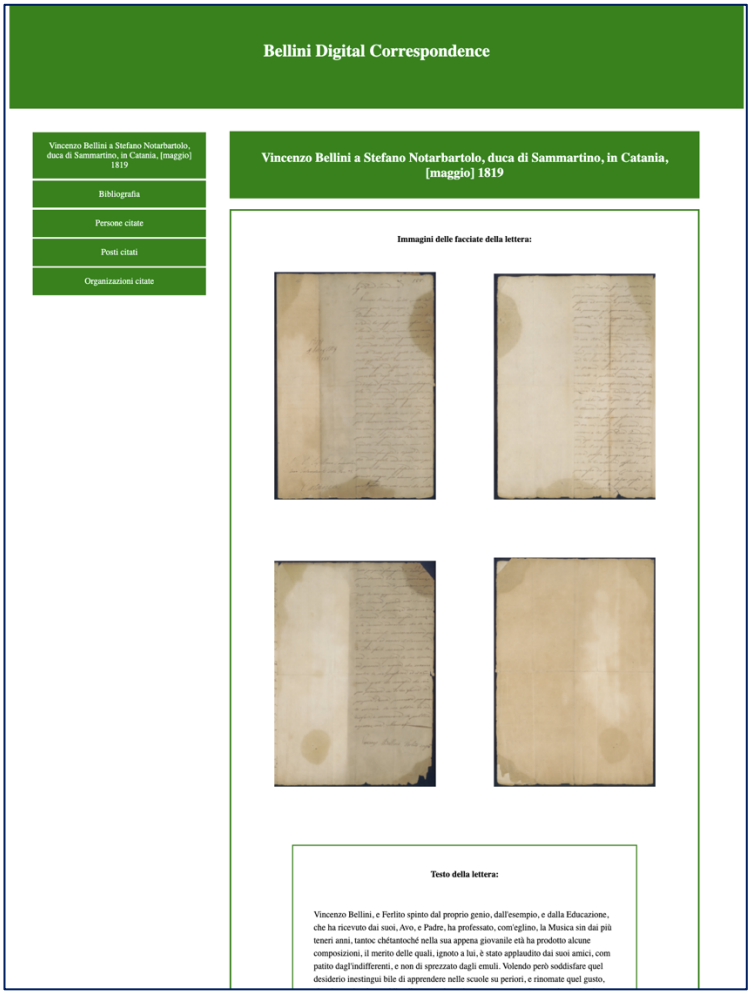

Figura 16. Applicazione realizzata usando Saxon-js che consente di visualizzare le informazioni sulla lettera LL1.1.

8.6. XQuery

XQuery (XML Query Language) è un linguaggio di programmazione realizzato e mantenuto dal W3C progettato allo scopo di interrogare documenti XML.

XQuery, sviluppato parallelamente a XSLT, è stato concepito per estrarre informazioni da uno o più documenti XML e, come XSLT, anch'esso può trasformare documenti XML da un formato all'altro.

XQuery oltre ad estrarre informazione dai documenti XML permette anche di eseguire numerose operazioni di manipolazioni ed analisi. Consente infatti di selezionare, filtrare, riorganizzare, ordinare, raggruppare, elaborare ed eventualmente trasformare i dati di interesse contenuti in collezioni di documenti XML al fine di generare documenti - in vari formati - con le informazioni richieste.

Tale linguaggio di interrogazione, basato su espressioni XPath, è supportato da tutti i principali database XML e può essere usato per:

- estrarre informazioni al fine di utilizzarle in un servizio web;
- generare report sui dati archiviati per la presentazione sul web;
- trasformare i dati nativi XML in XHTML;
- ricercare informazioni all'interno di un documento o di un insieme di documenti.

La struttura di base delle istruzioni per la scrittura di query è l'espressione *FLWOR*: un acronimo dove ogni lettera è l'iniziale di una clausola per la realizzazione delle istruzioni di interrogazione. Le clausole in oggetto sono: *For, Let, Where, Order by e Return*.

I *FLWOR*, a differenza delle espressioni XPath, consentono di manipolare, trasformare e ordinare i risultati.

- La clausola *For* si usa per selezionare una sequenza di nodi;
- la clausola *Let* serve per collegare una sequenza a una variabile;
- la clausola *Where* filtra i nodi che soddisfano una condizione booleana;
- la clausola *Order by* dispone i nodi specificando l'ordinamento del risultato;
- la clausola *Return* restituisce il risultato dell'interrogazione.

8.7. Esecuzione di XQuery mediante il processore Saxon

Lo strumento Saxon, usato precedentemente come processore XSLT, può essere usato anche per eseguire interrogazioni XQuery.[76] Infatti, Saxon utilizza lo stesso motore per eseguire sia XQuery che XSLT, riflettendo il fatto che i due linguaggi hanno una semantica molto simile ed anche la maggior parte del codice in fase di compilazione è comune.

Il processore XQuery può essere richiamato dalla riga di comando del sistema operativo o tramite un'API da un'applicazione sviluppata dall'utente.

Un esempio di riga di comando per eseguire una query con il processore Saxon scritto in Java è il seguente:

[76] Vedi <https://www.saxonica.com/html/documentation10/using-xquery/>.

```
java net.sf.saxon.Query -q:source.xql -o:index.html
```

L'opzione `-q` identifica il file contenente la query da eseguire, l'opzione `-o` registra l'output nel file indicato. In alternativa esiste anche l'opzione `-qs` che consente di inserire la query direttamente nella riga di comando.

8.8. Sperimentazione ed utilizzo di Saxon come processore XQuery per la creazione della vista web per l'edizione *Bellini Digital Correspondence*

Grazie al processore Saxon è stato possibile sperimentare la creazione di pagine web per la pubblicazione dell'edizione *Bellini Digital Correspondence* tramite XQuery permettendo di inserire le informazioni di interesse attraverso l'interrogazione dei file XML con query definite in documenti in formato XQL (vedere listati nella sezione 13.3 dell'appendice).

Questi file consentono di creare le diverse pagine web tra cui:

- la pagina *lettera.html* che mostra le immagini, il testo e le relative informazioni sul supporto e sulle altre mani della prima missiva che fa parte del *corpus* dell'edizione digitale (vedi figure 18 e 29);
- la pagina *bibliografia.html* che elenca la bibliografia di riferimento dell'intero *corpus* belliniano (vedi figure 19 e 32);
- le pagine *listapersone.html*, *listaposti.html* e *listaorg.html* che riportano l'elenco delle persone, dei luoghi e delle organizzazioni citate nel *corpus* (vedi figure 20, 28, 30 e 31).

Le informazioni di interesse da inserire in ogni pagina web sono state estratte dai documenti XML delle lettere di Bellini codificate per il progetto *Bellini Digital Correspondence*. Ad esempio, per visualizzare la lista delle persone citate nell'edizione digitale è stata usata la seguente la query:

```
[…]
<div class="list">
  <ul>{
    for $testo in doc("lists/TEI-
    ListPerson.xml")//tei:listPerson//tei:person/tei:persName
    return <li>{$testo//text()}</li>}</ul>
</div> […]
```

Per visualizzare la bibliografia invece è stata usata la seguente query:

```
[…]
<div class="biblio"><ul>{
    for $biblio in doc("TEI-ListBibl.xml")//tei:biblStruct
    return <li>
    {$biblio//tei:author//text()|$biblio//tei:editor//text()}
    . ({$biblio//tei:date//text()}).
    {$biblio//tei:title//text()}.
    {$biblio//tei:publisher//text()}.</li>}</ul>
</div>
[…]
```

Per visualizzare le immagini e per il testo della prima missiva:

```
[…]
<div>
    <img src="images/LL1-1_0001.jpg" width="300" alt=”LL1-
    1.1”/>
    <img src="images/LL1-1_0002.jpg" width="300" alt=”LL1-
    1.2”/>
    <img src="images/LL1-1_0003.jpg" width="300" alt=”LL1-
    1.3”/>
    <img src="images/LL1-1_0004.jpg" width="300" alt=”LL1-
    1.4”/>
</div>
<div id="text1" class="text-left" >
    <b>Testo della lettera:</b><p>{
    for $testo in doc("letter/LL1_1.xml")//tei:text//tei:body
    return $testo//tei:div[@type='letter-body']//text()}</p>
</div>
[…]
```

Una volta implementate le interrogazioni XQuery vengono ricavate le relative pagine web corrispondenti eseguendo su terminale le righe di comando (v. listato nella sezione 13.3 dell’appendice).

Di seguito un esempio di riga di comando:

```
java -cp saxon-he-10.3.jar net.sf.saxon.Query -q:lettera.xql -o:lettera.html
```

Ottenendo così le varie pagine sperimentate per la vista web dell'edizione digitale *Bellini Digital Correspondence* (vedi figure 18, 19, 20, 28, 29, 30, 31 e 32).

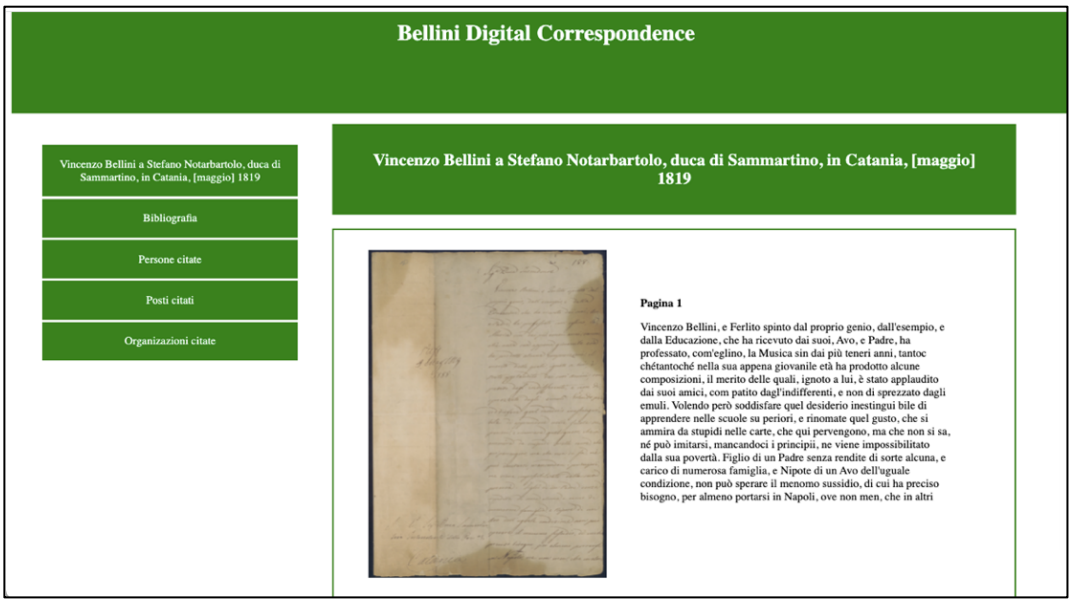

Figura 18. Pagina dell'applicazione realizzata usando X-Query che consente di visualizzare le informazioni sulla lettera LL1.1.

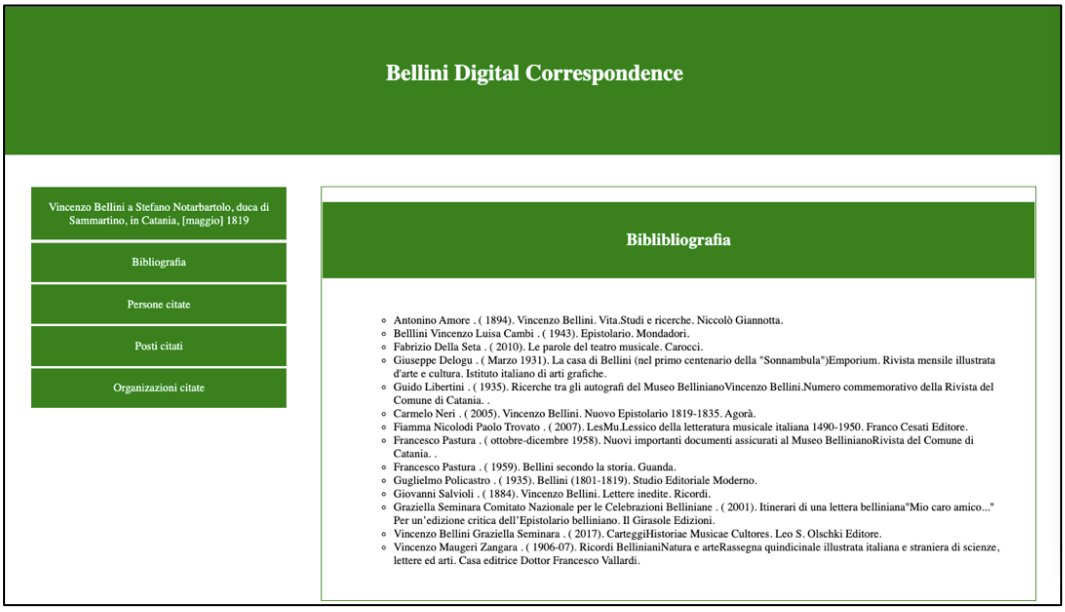

Figura19. Pagina dell'applicazione realizzata con XQuery che consente di visualizzare le informazioni sulla bibliografia.

Figura 20. Applicazione realizzata usando XQuery che consente di visualizzare le persone nominate nell'edizione digitale.

8.9. eXist-db

eXist-db è un software open source scritto in Java che fa uso del linguaggio XQuery di interrogazione standardizzato sviluppato dal W3C. Grazie a questo software è quindi possibile interrogare database basati su tecnologia XML.

eXist-db, inoltre, ha anche altre funzionalità e può essere usato come:

- motore di ricerca di documenti;
- piattaforma di sviluppo e messa in produzione di applicazioni web per la creazione ed acquisizione di documenti;
- web server per richiedere direttamente al database di archiviare, recuperare o aggiornare i documenti XML che contiene.

Quest' ultima operazione è possibile grazie alla disponibilità di una API REST HTTP[77] che espone le funzionalità di accesso al database.

Quando si esegue eXist-db come server, è possibile sviluppare intere applicazioni utilizzando linguaggi di trasformazione e di interrogazione come XSLT e XQuery.

[77] Il termine REST (Representational State Transfer) rappresenta un sistema di trasmissione di dati tra client e server basato sul protocollo HTTP.
I sistemi di trasmissione dei dati utilizzano i verbi (GET, POST, PUT, DELETE) del protocollo HTTP per recuperare o modificare uno o più documenti presenti sul server e identificati attraverso un url.
Su eXist-db le risorse sono accessibili all'url <*http://localhost:8080/exist/rest/db/BDC/*> solo se l'applicazione è eseguita sul database in locale.

eXist-db è soprattutto un XML native database e fornisce un potente ambiente per lo sviluppo di applicazioni web usando standard W3C quali XQuery, XSLT, XHTML, CSS e Javascript (con tecniche di comunicazione client-server asincrone); esso archivia e gestisce file XML secondo il modello gerarchico degli stessi nativo. Quindi, la struttura del database è determinata dalla struttura della collezione XML dei dati archiviati.

Operare con il programma è molto semplice e una volta installato basta configurarlo al primo avvio. L'interfaccia del programma permette di attivare tutte le funzionalità che il software offre. Si può ad esempio avviare il server, spegnerlo, aprire la dashboard[78] ed aprire eXide.

eXide è un editor integrato con eXist-db che permette di lavorare con XQuery, XML, HTML ed altre risorse archiviate nel database. Comprende molte funzionalità e permette di caricare, scaricare, visualizzare e modificare i documenti salvati nel database.

8.10. Implementazione del prototipo di edizione tramite l'impiego del software eXist-db

eXist-db consente di realizzare una vista web per la presentazione dei dati dell'edizione digitale BDC grazie alla creazione di un'applet.[79] Fino alla versione 4 di eXist-db era possibile creare un'applicazione direttamente da eXide.

Questa funzione purtroppo è stata rimossa dalla corrente versione del database (versione 5) impedendoci di poter creare l'applicazione direttamente su eXist-db.

La modalità ora suggerita per la creazione dell'applet è tramite l'utilizzo del tool Yeoman[80] in combinazione con il generatore di applet di eXist-db.[81]

---

[78] Apre l'url <http://localhost:8080/exist/apps/dashboard/> e mostra il pannello di controllo di eXist con tutte le applicazioni installate.

[79] Un applet è un programma progettato per essere eseguito all'interno di un altro programma-contenitore, in questo caso sull'interfaccia di eXist-db.

[80] Yeoman è un tool NPM che consente di creare automaticamente una struttura di base per un'applicazione o un progetto grazie all'utilizzo di generatori, ovvero di plugin. <https://yeoman.io/>.

[81] <https://github.com/eXist-db/generator-exist>.

Yeoman è un ambiente esterno ad eXist-db che permette di creare la struttura di base dell'applet. Yeoman si può installare usando Node.js tramite NPM con la riga di comando: `sudo npm i -g yo`.

Il generatore invece si installa col comando `sudo npm i -g @existdb/generator-exist`.

A questo punto è possibile eseguire il comando per l'inizializzazione dell'applet *BDC* generando la cartella con tutti i file di default.

All'interno dell'applet sono state aggiunte le immagini create utilizzando il software Adobe Photoshop, la cartella *Lettere* in cui sono contenuti i documenti XML di tutte le lettere del *corpus* e la cartella Lists con i documenti XML delle entità nominate nell'edizione digitale (quali persone, loghi, bibliografia, opere).

Successivamente, per personalizzare l'applicazione, le risorse sono state caricate all'interno del database nella directory Apps attraverso lo strumento eXide usando il menù principale (v. figura 21).

A questo punto è possibile personalizzare ed implementare i file dell'applicazione.

Una volta completate tutte le modifiche, alla fine del lavoro per poter caricare e visualizzare l'applet su eXist-db occorre eseguire alcuni ulteriori passaggi.

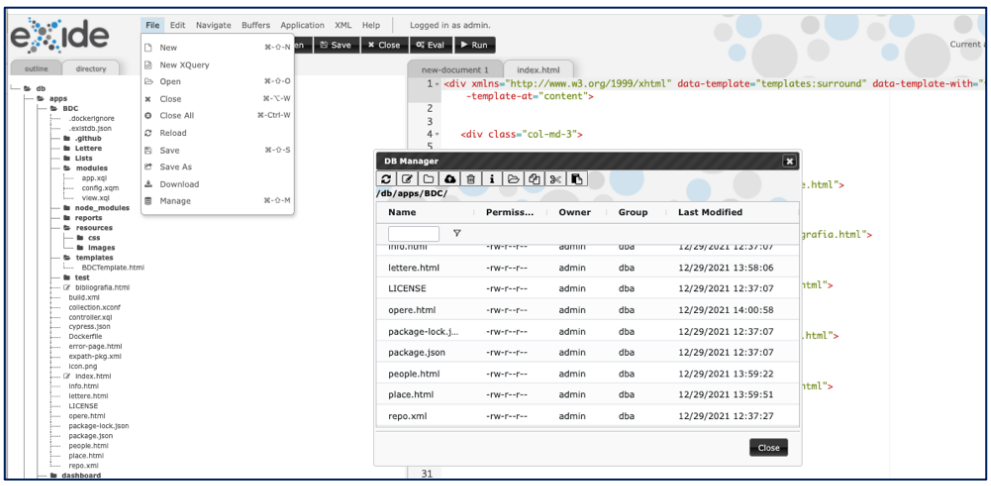

Figura 21. Caricamento Cartella Applet su eXide.

Dapprima si deve scaricare l'applicazione selezionando dal menu *Application* la voce *Download App*. Si scaricherà un file in formato compresso con estensione.*xar* che andrà poi caricato sulla Dashboard di Exist-db, selezionando dal menù laterale la sezione *Package manager* e poi cliccando sul pulsante *upload* (v. figura 22).

Una volta caricata, l'applicazione sarà visibile nella sezione Launcher.

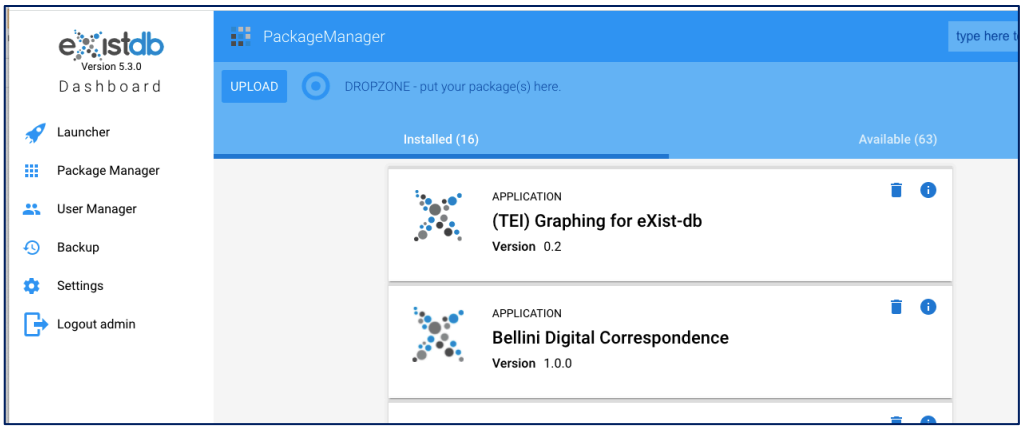

Figura 22. Caricamento dell'Applet su eXist-db.

Di seguito, in figura 23, lo schema che mostra i passaggi eseguiti per la creazione dell'applet su eXist-db.

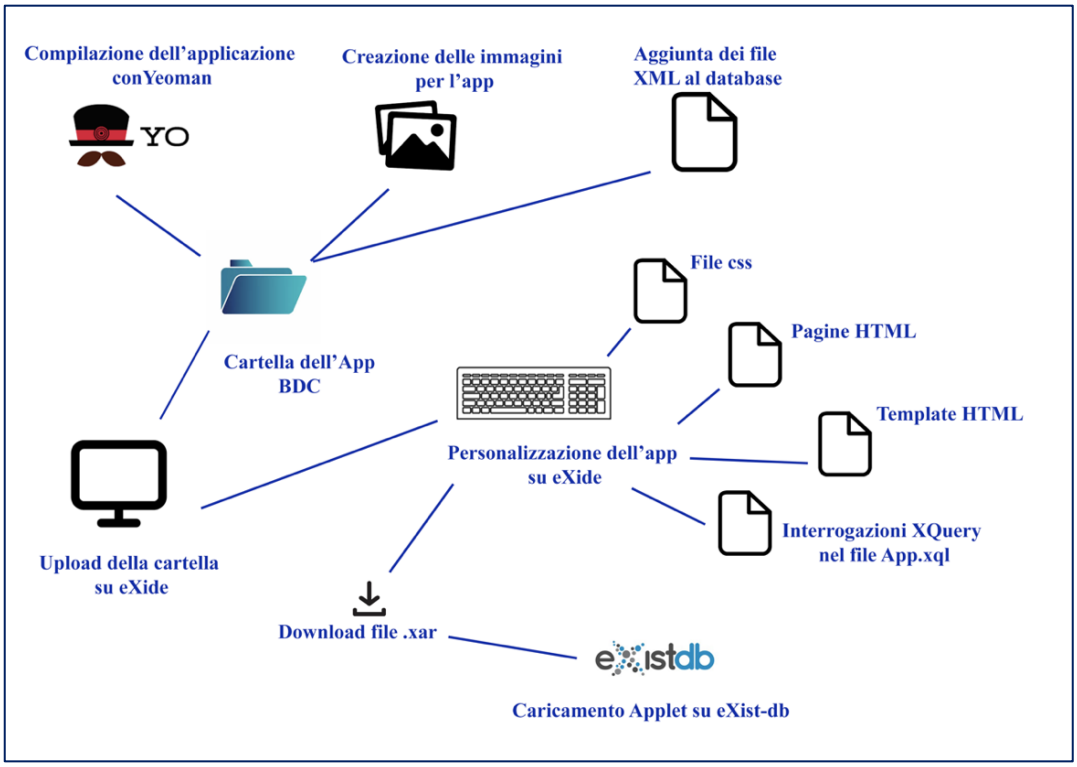

Figura 23. Processo di creazione dell'applet su eXist-db.

L'applicazione creata mostra sei pagine web in cui si possono visualizzare le informazioni relative alla bibliografia, alle lettere che compongono il *corpus*, alle opere, alle persone e ai luoghi citati nell'edizione.

Per realizzare l'applicazione alcuni file presenti nella cartella sono rimasti invariati mentre altri sono stati modificati. È stata personalizzata la pagina *index.html* e create le pagine HTML a cui si collegano i link presenti sull'interfaccia di fruizione, tra cui la pagina della bibliografia, delle persone, delle opere e dei luoghi citati nell'edizione e delle lettere presenti nel *corpus* (vedi figure 24, 33, 34, 35, 36, 37, 38 e 39). Successivamente sono state implementate le regole del foglio di stile CSS contenuto nella cartella *resources*, la pagina HTML di template collocata all'interno della cartella *templates* e il file *app.xql* posizionato nella directory *modules* (vedere listati nella sezione 13.4 dell'appendice).

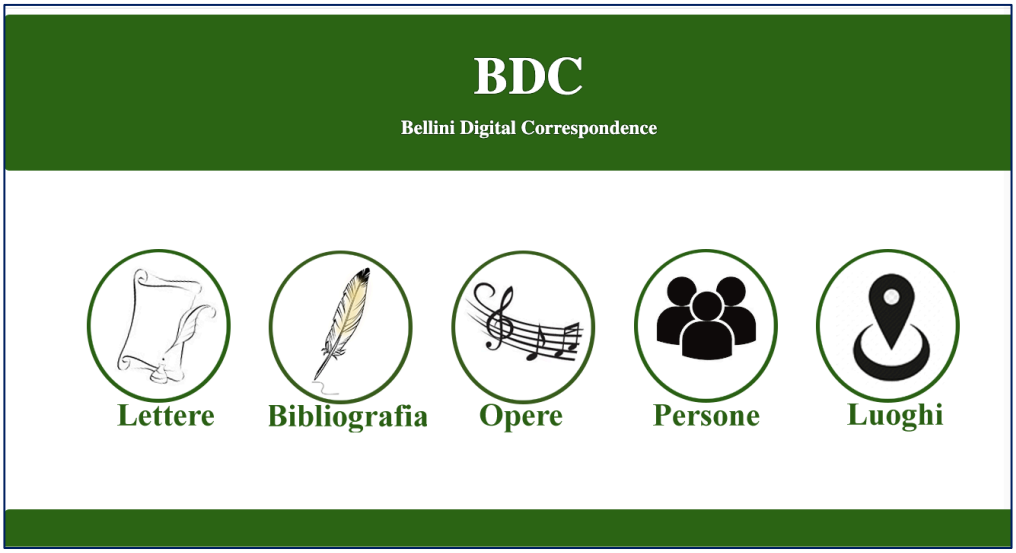

Figura 24. Interfaccia della homepage dell'applet su eXist-db (*Index.html*).

La pagina HTML *BDCtemplate.html* è un modello di pagina HTML contenente tutte le parti ripetute del layout HTML. Questa funzionalità che permette di impostare dei template per le pagine web è molto interessante per la creazione dell'applicazione.
Il file *app.xql* comprende la libreria di funzioni contenenti le interrogazioni XQuery implementate. Quest'ultime consentono di estrarre e visualizzare le informazioni sulla pagina (v. listato *app.xql* nella sezione 13.3 dell'appendice). Ogni funzione si può richiamare utilizzando l'attributo *custom*[82] `@data-template` che permette di inserire il risultato ottenuto dalla query nell'elemento in cui compare questo attributo.

[82] Un attributo *custom* (chiamato anche *data-*)* viene usato per archiviare dati personalizzati nella pagina o nell'applicazione Web. Quest'attributo si può incorporare in tutti gli elementi HTML.

Un esempio di porzione di codice HTML:

```
<div id="biblio" data-template="app:biblio"/>
```

Un esempio di interrogazione Xquery inclusa nel file app.xql:

```
declare function app:biblio($node as node(), $model as map(*))
{for $biblio in doc("/db/apps/BDC/Lists/TEI-
ListBibl.xml")//tei:biblStruct

return <ul><li>
{$biblio//tei:author//text()|$biblio//tei:editor//text()}. (
{$biblio//tei:date//text()}).

{$biblio//tei:title//text()}.
{$biblio//tei:publisher//text()}. </li></ul>};
```

Il risultato è la pagina che è mostrata nella figura 25:

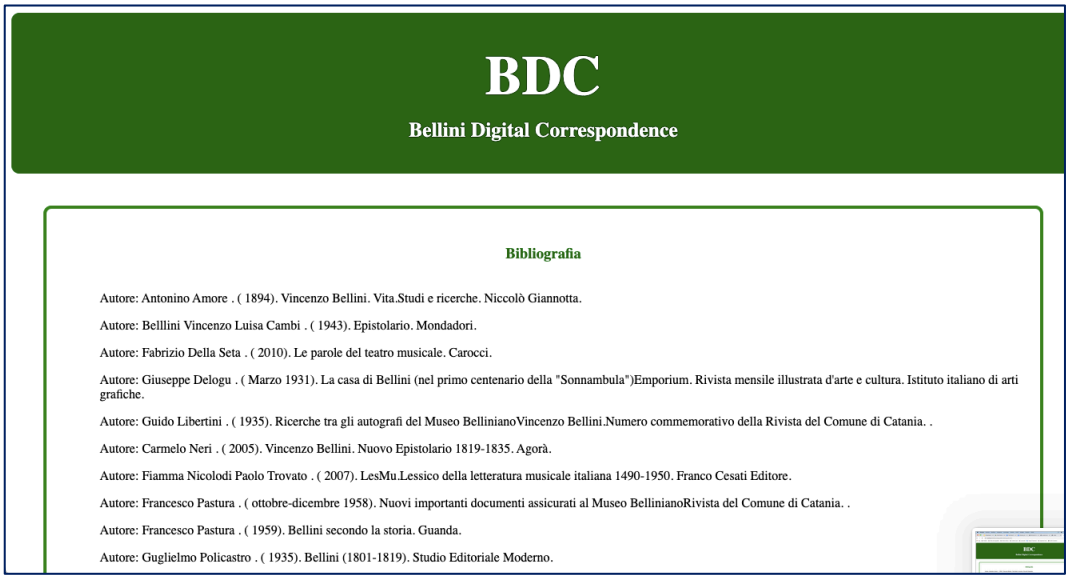

Figura 25. Pagina *bibliografia.html* dell'applet su eXist-db.

<https://www.w3schools.com/tags/att_data-.asp>.

9. Conclusioni e sviluppi futuri

Il lavoro svolto per il progetto di tesi, descritto nei vari capitoli dell'elaborato, ha avuto come principale obiettivo quello di ottimizzare la presentazione e la fruizione dell'edizione digitale delle lettere autografe di Bellini conservate presso il Museo Civico Belliniano di Catania. Sono state descritte puntualmente le varie fasi del processo di edizione a cui è stato sottoposto il repertorio di codifica al fine di poter pubblicare le informazioni e i dati mediante l'uso del software EVT.

A tale scopo, i dati presenti nei documenti di codifica XML, che pur essendo codificati seguendo le linee guida della Text Encoding Initiative, non sono perfettamente aderenti alle specifiche di elaborazione di EVT. Conseguentemente, i documenti delle singole lettere sono stati manipolati e trasformati attraverso l'applicazione di numerose regole di trasformazione XSLT tramite il processore XSLT Saxon.

Il lavoro di adattamento della codifica e di ristrutturazione dei documenti XML ha prodotto un ottimo risultato, ma per colmare la difficoltà dell'approccio alla corretta codifica delle informazioni ed avere maggiore libertà nell'utilizzo dei dati del *corpus* (rappresentati tramite gli elementi definiti dal vocabolario TEI), l'attività di ricerca tecnologica e di sviluppo si è spostata sulla sperimentazione di nuove soluzioni prendendo in considerazione l'utilizzo di ulteriori tecnologie XML-based finalizzate alla pubblicazione dell'edizione digitale. Inoltre, è risultata molto flessibile l'idea di poter realizzare presentazioni web personalizzabili senza dover ricorrere a complesse operazioni di trasformazione, di elaborazione e di implementazione dei file di codifica iniziali.

Le applicazioni create sono state depositate e rese pubbliche con licenza MIT al seguente link: https://github.com/santapellino/Progetto-tesi-Santa-Pellino.

L'edizione digitale *Bellini Digital Correspondence* è consultabile on line sui siti: http://bellinicorrespondence.cnr.it/ e http://licodemo.ilc.cnr.it/bellini-in-rete/.

Il sito del progetto ufficiale è invece raggiungibile al seguente link: http://bellininrete.istc.cnr.it/index.html.

In conclusione, la sperimentazione di queste nuove tecnologie ha dimostrato che, anche se EVT è un software molto valido per la fruizione delle edizioni digitali, esistono approcci e tecnologie integrative per l'elaborazione e la presentazione dei dati digitali.

Difatti il lavoro di tesi ha dimostrato come possiamo sfruttare diverse tecnologie se realizziamo una corretta rappresentazione dei dati, i quali poi possono essere presentati all'utente finale adottando uno spettro ampio di strumenti quali Saxon-Js, XQuery ed eXist-db con cui è possibile ottenere buoni risultati.

Grazie alla tecnologia Saxon-JS è stato possibile realizzare un'applicazione web che ci permetta di eseguire l'elaborazione dei template XSLT direttamente all'interno di un browser Web, eseguendo, in questo modo, solo il codice di Run-time e rendendo così il processo di esecuzione molto più veloce.

Con la tecnologia XQuery invece è possibile ottenere viste web grazie all'interrogazione dei dati contenuti nei documenti XML attraverso la definizione di query specificate in file in formato XQL. L'applicazione è poi realizzabile sia attraverso l'utilizzo del processore Saxon, sia attraverso l'impiego del software eXist-db. Il valore aggiunto del processo di edizione è quindi individuabile nella corretta rappresentazione della fonte primaria, poiché, se ben strutturata e progettata, consente di presentare ed elaborare i dati mediante un ampio spettro di tecnologie.

Le indagini e gli sviluppi realizzati durante il lavoro di tesi hanno prodotto interessanti risultati sia nel merito del progetto *Bellini Digital Correspondence* sia per l'utilizzo di metodi e tecnologie nuove anche ai membri del team di lavoro più esperti.

In ultimo, i risultati ottenuti hanno aperto le strade per numerosi sviluppi futuri. Tra questi:

- l'ampliamento del *corpus* dell'edizione digitale aggiungendo, oltre alle lettere scritte da Bellini, anche le altre lettere conservate nel Museo Civico Belliniano di Catania, come per esempio quelle destinate a Bellini;
- l'estensione dell'applicazione creata con la tecnologia Saxon-JS in modo da poter includere anche le altre lettere del *corpus* e non solo la prima missiva;
- l'arricchimento dei dati codificati con, ad esempio, l'inserimento dei dati emersi da un'analisi di tipo linguistico, stilistico o di scrittura (destinatari, luoghi, date, mittenti);
- l'estensione dell'applicazione creata per il software eXist-db in modo che includa anche la visualizzazione del testo delle lettere;
- un ulteriore sviluppo dell'elaborazione della codifica dei dati con l'aggiunta e l'approfondimento tecnologico dei termini musicali, tra cui il linguaggio del lessico della letteratura musicale italiana (LESMU).

10. Ringraziamenti

A conclusione di questo elaborato desidero dedicare qualche riga a coloro che hanno contribuito alla realizzazione della mia tesi e a coloro che mi sono stati vicini e mi hanno sostenuta fino al raggiungimento di questo importante obiettivo.

Innanzitutto, ringrazio la mia relatrice Marina Riccucci e i miei correlatori Angelo Mario Del Grosso e Daria Spampinato per la disponibilità, per avermi guidato nella ricerca e nella realizzazione dell'elaborato e per aver contribuito alla mia crescita personale e professionale.

Vorrei ringraziare Pietro Sichera e Laura Mazzagufo per i consigli e l'aiuto ricevuto durante la fase di ricerca e di realizzazione del mio lavoro.

Un ringraziamento speciale a Matteo: è anche grazie al suo amore, al suo sostegno e al suo incoraggiamento che sono riuscita a raggiungere questo traguardo.

Una dedica speciale va alla mia famiglia e ai miei amici, in particolare a Giulia, che mi sono stati vicini e che hanno condiviso con me sacrifici e gioie. Grazie per l'affetto e il sostegno che mi avete dato.

11. Bibliografia

Burnard, Lou, e Michael Sperberg-McQueen. 2021. *TEI P5: Guidelines for Electronic Text Encoding and Interchange*.

Cambi, Luisa. 1943. *Bellini. Epistolario.* Milano: Mondadori.

Cambi, Luisa. 1973. *Bellini. Un pacchetto di autografi, in Scritti in onore di Luigi Ronga.* Milano-Napoli: Ricciardi.

Capizzi, Erica, Salvatore Cristofaro, Maria Rosa De Luca, Del Grosso Angelo Mario, Emiliano Giovannetti, Marchi Simone, Graziella Seminara, e Daria Spampinato. 2019. *Bellini's Correspondence: a Digital Scholarly Edition for a Multimedia Museum. Umanistica Digitale.* https://umanisticadigitale.unibo.it/article/view/9162

Coletti Vittorio, Vittorio. 2017. «*Vincenzo Bellini – Carteggi. Silenzio universale, applausi unanimi.*». L'indice.

Cristofaro, Salvatore, Maria Rosa De Luca, Angelo Mario Del Grosso, Emiliano Giovannetti, Simone Marchi, Graziella Seminara, e Daria Spampinato. *2018. AIUCD 2018 - Book of Abstracts, edited by Daria Spampinato, 60-64. Le Lettere Di Bellini: Dalla Carta al Web*.

Doug, Tidwell. 2008. *XSLT: mastering XML transformation.* United States of America: O'Reilly Media.

Florimo, Francesco. 1882. *Bellini. Memorie e lettere.* Firenze: Barbèra.

Florimo, Francesco. 1969. *Cenno storico sulla scuola musicale di Napoli.* 2 voll. Napoli: Rocco.

John, Rosselli. 2004. «Per un'edizione critica dell'epistolario belliniano, in Vincenzo Bellini. Verso l'edizione critica.» In *Atti del Convegno internazionale (Siena, 1-3*

giugno 2000), a cura di Fabrizio Della Seta e Simonetta Ricciardi, 292. Firenze: Olschki.

Lippmann, Friedrich. 1977. *Belliniana, in Il melodramma italiano dell'Ottocento. Studi in onore di Massimo Mila.* Torino: Einaudo.

Neri, Carmelo. 2001. *Caro Bellini... Lettere edite e inedite a Vincenzo Bellini.* Catania: Prova d'autore.

Neri, Carmelo. 1991. *Lettere di Vincenzo Bellini (1819-1835)*. Catania: Publisicula.

Neri, Carmelo. 2005. *Vincenzo Bellini. Nuovo Epistolario. 1819-1835.* Catania: Agorà.

Pastura, Francesco. 1959. *Bellini secondo la storia.* Parma: Guanda.

Pastura, Francesco. 1935. *Le lettere di Bellini (1819-1835). Prima edizione integrale raccolta, ordinata e annotata da Francesco Pastura.* Catania: Totalità.

Retter, Adam, e Erik Siegel. 2014. *EXist: A NoSQL Document Database and Application Platform.* United States of America: O'Reilly Media.

Seminara, Graziella. 2017. *Vincenzo Bellini. Carteggi.* Firenze: Olschki.

Toscani, Claudio. 2017. «*Vincenzo Bellini. Carteggi, edizione critica a cura di Graziella Seminara, Firenze, Olschki, 2017.»* Bollettino di Studi Belliniani.

Walmsley, Priscilla. 2015. *XQuery: Search Across a Variety of XML Data.* 2ª ed. United States of America: O'Reilly Media.

Walker, Frank. 1960. «Lettere disperse e inedite di Vincenzo Bellini.» *Rivista del Comune di Catania*, dicembre 1960.

12. Sitografia

https://amsacta.unibo.it/5997/1/AIUCD-2018-BoA-rev.pdf

http://bellinicorrespondence.cnr.it/

http://bellininrete.istc.cnr.it/

https://doi.org/10.6092/issn.2532-8816/9162

http://evt.labcd.unipi.it/

http://exist-db.org/exist/apps/doc/

https://github.com/

https://github.com/eXist-db/generator-exist

https://it.wikipedia.org/wiki/Museo_civico_belliniano

https://it.wikipedia.org/wiki/Text_Encoding_Initiative

https://it.wikipedia.org/wiki/Vincenzo_Bellini

https://it.wikipedia.org/wiki/XML

https://it.wikipedia.org/wiki/XSL_Transformations/

http://licodemo.ilc.cnr.it/bellini-in-rete/

http://saxon.sourceforge.net/

https://tei-c.org/

https://tei-c.org/guidelines/p5/

https://tei-c.org/release/doc/tei-p5-doc/en/html/SG.html

https://umanisticadigitale.unibo.it/article/view/9162/9914

http://xmlsoft.org/xmllint.html

https://yeoman.io/

https://www.algraeditore.it/autore/carmelo-neri/

https://www.beniculturali.it/luogo/museo-civico-belliniano

http://www.cafeconleche.org/books/bible2/chapters/ch17.html

http://www.labcd.unipi.it/progetti/evt-edition-visualization-technology/

https://www.lindiceonline.com/focus/musica/vincenzo-bellini-carteggi/

https://www.saxonica.com/

https://www.saxonica.com/html/documentation10/using-xquery/

https://www.saxonica.com/saxon-js/documentation2/

https://www.saxonica.com/saxon-js/documentation2/index.html#!about/components

https://www.saxonica.com/saxon-js/documentation2/index.html#!browser/result-documents

https://www.saxonica.com/saxon-js/documentation2/index.html#!ixsl-extension/instructions

https://www.saxonica.com/technology/xslt-and-xquery.xml#xslt

https://www.w3.org/TR/xinclude-11/

https://www.w3.org/TR/xslt-30/

https://www.w3schools.com/xml/xquery_intro.asp

13. Appendice

13.1. Template XSLT

I template XSLT sono collocati nel file *bellini.xsl.*

Di seguito il codice del template implementato per l'elemento `surface`.

```
<xsl:template match="tei:surface">
    <xsl:variable name="idno"
    select="//tei:idno[@type='inventory']"/>
    <xsl:variable name="n" select="@n"/>
    <xsl:variable name="newN" select="concat ($idno, '.',
    $n)"/>
    <xsl:variable name="id" select="@xml:id"/>
    <xsl:variable name="pb" select="//tei:body//tei:pb"/>
    <xsl:element  name="surface"
    namespace="http://www.tei-c.org/ns/1.0">
      <xsl:attribute name="n">
        <xsl:copy-of select=" $newN"/>
      </xsl:attribute>
      <xsl:attribute name="xml:id">
        <xsl:copy-of select="$id"/>
      </xsl:attribute>
      <xsl:attribute name="corresp">
        <xsl:for-each select="$pb">
          <xsl:if test="@n=$n">
            <xsl:variable name="facs" select="@xml:id"/>
            <xsl:variable name="newfacs" select="concat('#',
             $facs)"/>
            <xsl:value-of select="$newfacs"/>
          </xsl:if>
        </xsl:for-each>
      </xsl:attribute>
      <xsl:apply-templates select="tei:graphic"/>
      <xsl:variable name="width" select="tei:graphic/@width"/>
      <xsl:variable name="widthnew" select="substring-
      before($width, 'px')"/>
```

```
<xsl:variable name="coeff" select="number(723)"/>
<xsl:variable name="rapp" select="number($widthnew) div
 $coeff"/>
<xsl:for-each select="tei:zone ">
  <xsl:variable name="id" select="@xml:id"/>
  <xsl:variable name="corresp" select="concat ('#', $id,
   'h')"/>
  <xsl:variable name="rendition" select="@rendition"/>
  <xsl:variable name="ulx" select="@ulx"/>
  <xsl:variable name="uly" select="@uly"/>
  <xsl:variable name="lrx" select="@lrx"/>
  <xsl:variable name="lry" select="@lry"/>
  <xsl:element  name="zone" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:attribute name="ulx">
      <xsl:variable name="result" select="$ulx div
       $rapp"/>
      <xsl:value-of select="format- number($result,
       '#.00')"/>
    </xsl:attribute>
    <xsl:attribute name="uly">
      <xsl:variable name="result" select="$uly div
       $rapp"/>
      <xsl:value-of select="format- number($result,
       '#.00')"/>
    </xsl:attribute>
    <xsl:attribute name="lrx">
      <xsl:variable name="result" select="$lrx div
       $rapp"/>
      <xsl:value-of select="format-number($result,
       '#.00')"/>
    </xsl:attribute>
    <xsl:attribute name="lry">
      <xsl:variable name="result" select="$lry div
       $rapp"/>
      <xsl:value-of select="format- number($result,
       '#.00')"/>
```

```
      </xsl:attribute>
      <xsl:attribute name="rendition">
        <xsl:value-of select="$rendition"/>
      </xsl:attribute>
      <xsl:attribute name="xml:id">
        <xsl:value-of select="$id"/>
      </xsl:attribute>
      <xsl:if test="$rendition='HotSpot'">
        <xsl:attribute name="corresp">
          <xsl:value-of select="$corresp"/>
        </xsl:attribute>
        <xsl:attribute name="start">
          <xsl:value-of select="$corresp"/>
        </xsl:attribute>
      </xsl:if>
    </xsl:element>
  </xsl:for-each>
  <xsl:if test="exists(tei:zone/tei:zone)">
    <xsl:for-each select="tei:zone/tei:zone">
      <xsl:variable name="id" select="@xml:id"/>
      <xsl:variable name="corresp" select="concat ('#',
       $id, 'h')"/>
      <xsl:variable name="rendition" select="@rendition"/>
      <xsl:variable name="ulx" select="@ulx"/>
      <xsl:variable name="uly" select="@uly"/>
      <xsl:variable name="lrx" select="@lrx"/>
      <xsl:variable name="lry" select="@lry"/>
      <xsl:element  name="zone" namespace="http://www.tei-
       c.org/ns/1.0">
        <xsl:attribute name="ulx">
          <xsl:variable name="result" select="$ulx div
           $rapp"/>
          <xsl:value-of select="format- number($result,
           '#.00')"/>
        </xsl:attribute>
        <xsl:attribute name="uly">
```

```
              <xsl:variable name="result" select="$uly div
               $rapp"/>
              <xsl:value-of select="format- number($result,
               '#.00')"/>
            </xsl:attribute>
            <xsl:attribute name="lrx">
              <xsl:variable name="result" select="$lrx div
               $rapp"/>
              <xsl:value-of select="format- number($result,
               '#.00')"/>
            </xsl:attribute>
            <xsl:attribute name="lry">
              <xsl:variable name="result" select="$lry div
               $rapp"/>
              <xsl:value-of select="format-number($result,
               '#.00')"/>
            </xsl:attribute>
            <xsl:attribute name="rendition">
              <xsl:value-of select="$rendition"/>
            </xsl:attribute>
            <xsl:attribute name="xml:id">
              <xsl:value-of select="$id"/>
            </xsl:attribute>
              <xsl:if test="$rendition='HotSpot'">
            <xsl:attribute name="corresp">
              <xsl:value-of select="$corresp"/>
            </xsl:attribute>
            <xsl:attribute name="start">
              <xsl:value-of select="$corresp"/>
            </xsl:attribute>
          </xsl:if>
        </xsl:element>
      </xsl:for-each>
    </xsl:if>
  </xsl:element>
</xsl:template>
```

Di seguito il codice del template implementato per l'elemento `msDesc`.

```
<xsl:template match="tei:msDesc">
    <xsl:copy>
      <xsl:apply-templates select="tei:msIdentifier"/>
      <xsl:element name="msContents"
       namespace="http://www.tei-c.org/ns/1.0">
        <xsl:element name="msItem" namespace="http://www.tei-
         c.org/ns/1.0">
          <xsl:attribute name="xml:id">
            <xsl:text>msItem-</xsl:text>
            <xsl:value-of
             select="//tei:idno[@type='inventory']"/>
          </xsl:attribute>
            <xsl:element name="idno"
             namespace="http://www.tei-c.org/ns/1.0">
              <xsl:attribute name="type">inventory
              </xsl:attribute>
              <xsl:value-of
               select="//tei:idno[@type='inventory']"/>
            </xsl:element>
            <xsl:element name="title"
             namespace="http://www.tei-c.org/ns/1.0">
              <xsl:value-of
               select="//tei:titleStmt/tei:title"/>
            </xsl:element>
            <xsl:element name="bibl"
             namespace="http://www.tei-c.org/ns/1.0">
              <xsl:element name="fw"
               namespace="http://www.tei-c.org/ns/1.0">
                <xsl:text>Bibliografia </xsl:text>
              </xsl:element>
              <xsl:text>Seminara, 2017, </xsl:text>
              <xsl:element name="citedRange"
              namespace="http://www.tei-c.org/ns/1.0">
```

```
    <xsl:attribute
     name="source">#Seminara2017</xsl:attribute>
    <xsl:attribute name="resp">#GS</xsl:attribute>
    <xsl:attribute name="ana">
      <xsl:value-of
       select="//tei:sourceDesc//tei:bibl/@ana"/>
    </xsl:attribute>
    <xsl:element name="locus"
     namespace="http://www.tei-c.org/ns/1.0">
      <xsl:attribute name="from">
        <xsl:value-of
         select="//tei:sourceDesc//tei:bibl/tei:ci
         tedRange/@from"/>
      </xsl:attribute>
      <xsl:attribute name="to" >
        <xsl:value-of
         select="//tei:sourceDesc//tei:bibl/tei:ci
         tedRange/@to"/>
      </xsl:attribute>
      <xsl:text>n.</xsl:text>
      <xsl:text> </xsl:text>
      <xsl:value-of
       select="//tei:sourceDesc//tei:bibl/@ana"/>
      <xsl:text>,</xsl:text>
      <xsl:text> </xsl:text>
      <xsl:text>p.</xsl:text>
      <xsl:text> </xsl:text>
      <xsl:value-of
       select="//tei:sourceDesc//tei:bibl/tei:cite
       dRange"/>
      <xsl:text>.</xsl:text>
    </xsl:element>
  </xsl:element>
</xsl:element>
<xsl:variable name="biblio"
 select="//tei:back//tei:div[@type='bibliography']
 //tei:listBibl//tei:bibl//tei:ref//tei:bibl" />
```

```
<xsl:variable name="bibl2"
select="//tei:bibl//tei:ref//tei:bibl" />
<xsl:variable name="author"
 select="//tei:bibl//tei:ref//tei:bibl//tei:author
 " />
<xsl:variable name="date"
select="//tei:bibl//tei:ref//tei:bibl//tei:date"
/>
<xsl:variable name="pag"
select="//tei:bibl//tei:ref//tei:bibl//tei:citedRa
nge" />
  <xsl:if test="exists($author[1])">
    <xsl:element name="bibl"
    namespace="http://www.tei-c.org/ns/1.0">
      <xsl:value-of select="$author[1]"/>
      <xsl:text>, </xsl:text>
      <xsl:value-of select="$date[1]"/>
      <xsl:text>, p. </xsl:text>
      <xsl:value-of select="$pag[1]"/>
      <xsl:text>. </xsl:text>
    </xsl:element>
  </xsl:if>
  <xsl:if test="exists($author[2])">
    <xsl:element name="bibl"
    namespace="http://www.tei-c.org/ns/1.0">
      <xsl:value-of select="$author[2]"/>
      <xsl:text>, </xsl:text>
      <xsl:value-of select="$date[2]"/>
      <xsl:text>, p. </xsl:text>
      <xsl:value-of select="$pag[2]"/>
      <xsl:text>. </xsl:text>
    </xsl:element>
  </xsl:if>
  <xsl:if test="exists($author[3])">
    <xsl:element name="bibl"
    namespace="http://www.tei-c.org/ns/1.0">
      <xsl:value-of select="$author[3]"/>
```

```
      <xsl:text>, </xsl:text>
      <xsl:value-of select="$date[3]"/>
      <xsl:text>, p. </xsl:text>
      <xsl:value-of select="$pag[3]"/>
      <xsl:text>. </xsl:text>
    </xsl:element>
  </xsl:if>
  <xsl:if test="exists($author[4])">
    <xsl:element name="bibl"
    namespace="http://www.tei-c.org/ns/1.0">
      <xsl:value-of select="$author[4]"/>
      <xsl:text>, </xsl:text>
      <xsl:value-of select="$date[4]"/>
      <xsl:text>, p. </xsl:text>
      <xsl:value-of select="$pag[4]"/>
      <xsl:text>. </xsl:text>
    </xsl:element>
  </xsl:if>
  <xsl:element name="incipit"
   namespace="http://www.tei-c.org/ns/1.0" >
    <xsl:element name="fw"
     namespace="http://www.tei-c.org/ns/1.0">
      <xsl:text>Incipit </xsl:text>
    </xsl:element>
    <xsl:variable name="s1"
     select="//tei:s[@n='s_01']"/>
    <xsl:variable name="salute"
    select="//tei:div[@type='letter-
    body']//tei:salute"/>
     <xsl:choose>
       <xsl:when test="exists($s1)">
         <xsl:apply-templates select="$s1"
          mode="sciogliabbr"/>
       </xsl:when>
       <xsl:otherwise>
         <xsl:choose>
           <xsl:when test="exists($salute)">
```

```
                        <xsl:apply-templates select="$salute"
                         mode="sciogliabbr"/>
                      </xsl:when>
                      <xsl:otherwise>
                        <xsl:text>Incipit non
                         presente</xsl:text>
                      </xsl:otherwise>
                    </xsl:choose>
                  </xsl:otherwise>
                </xsl:choose>
              </xsl:element>
              <xsl:element name="explicit"
               namespace="http://www.tei-c.org/ns/1.0">
                <xsl:element name="fw"
                 namespace="http://www.tei-c.org/ns/1.0">
                  <xsl:text>Explicit </xsl:text>
                  </xsl:element>
                  <xsl:variable name="closer"
                   select="//tei:div[@type='closer']"/>
                  <xsl:choose>
                    <xsl:when test="exists($closer)">
                      <xsl:apply-templates select="$closer"
                       mode="sciogliabbr" />
                    </xsl:when>
                    <xsl:otherwise>
                      <xsl:text>Esplicit non
                       presente</xsl:text>
                    </xsl:otherwise>
                  </xsl:choose>
              </xsl:element>
        </xsl:element>
      </xsl:element>
    </xsl:copy>
</xsl:template>
```

Di seguito il codice del template implementato per l'elemento `front`.

```
<xsl:template match="tei:front">
    <xsl:element name="front" namespace="http://www.tei-
    c.org/ns/1.0">
       <xsl:element name="div" namespace="http://www.tei-
        c.org/ns/1.0">
         <xsl:attribute name="type">Title</xsl:attribute>
         <xsl:element name="fw" namespace="http://www.tei-
          c.org/ns/1.0">Titolo della lettera
         </xsl:element>
         <xsl:element name="p" namespace="http://www.tei-
          c.org/ns/1.0">
           <xsl:copy-of
            select="//tei:titleStmt//tei:title/node()"/>
         </xsl:element>
       </xsl:element>
       <xsl:element name="div" namespace="http://www.tei-
        c.org/ns/1.0">
         <xsl:attribute
          name="type">Bibliography</xsl:attribute>
         <xsl:element name="fw" namespace="http://www.tei-
          c.org/ns/1.0">Bibliografia</xsl:element>
         <xsl:element name="p" namespace="http://www.tei-
          c.org/ns/1.0">
            <xsl:text>Seminara 2017, n. </xsl:text>
            <xsl:value-of
             select="//tei:sourceDesc//tei:bibl/@ana"/>
            <xsl:text> </xsl:text>
            <xsl:text> p. </xsl:text>
            <xsl:value-of
             select="//tei:sourceDesc//tei:bibl/tei:citedRange
             "/>
          </xsl:element>
        </xsl:element>
```

```
<xsl:element name="div" namespace="http://www.tei-
 c.org/ns/1.0">
  <xsl:attribute name="type">language</xsl:attribute>
    <xsl:element name="fw" namespace="http://www.tei-
     c.org/ns/1.0">Lingua</xsl:element>
    <xsl:element name="p" namespace="http://www.tei-
     c.org/ns/1.0">
      <xsl:copy-of
       select="//tei:msItem/tei:textLang/node()"/>
      </xsl:element>
    </xsl:element>
    <xsl:element name="div" namespace="http://www.tei-
     c.org/ns/1.0">
      <xsl:attribute name="type">Collocation
      </xsl:attribute>
      <xsl:element name="p" namespace="http://www.tei-
       c.org/ns/1.0">
        <xsl:element name="fw"
         namespace="http://www.tei-c.org/ns/1.0">
          Collocazione
        </xsl:element>
        <xsl:value-of
         select="//tei:msIdentifier/tei:settlement"/>
        <xsl:text> (</xsl:text>
        <xsl:value-of
         select="//tei:msIdentifier/tei:country"/>
        <xsl:text>)</xsl:text>
        <xsl:text> - </xsl:text>
        <xsl:value-of
         select="//tei:msIdentifier/tei:repository/nod
         e()"/>
      </xsl:element>
      <xsl:element name="p" namespace="http://www.tei-
       c.org/ns/1.0">
        <xsl:value-of
         select="//tei:msIdentifier//tei:idno[@type='c
         ollocation']"/>
```

```
  </xsl:element>
  <xsl:element name="p" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:element name="fw"
     namespace="http://www.tei-c.org/ns/1.0">
      <xsl:text>Idno </xsl:text>
    </xsl:element>
    <xsl:value-of
     select="//tei:msIdentifier/tei:idno[@type='in
     ventory']"/>
  </xsl:element>
</xsl:element>
<xsl:element name="div" namespace="http://www.tei-
 c.org/ns/1.0">
    <xsl:attribute
     name="type">desc</xsl:attribute>
    <xsl:element name="head"
     namespace="http://www.tei-c.org/ns/1.0">
        Descrizione del support
    </xsl:element>
    <xsl:element name="p"
     namespace="http://www.tei-c.org/ns/1.0">
        <xsl:element name="fw"
         namespace="http://www.tei-c.org/ns/1.0">
          <xsl:text>Carta</xsl:text>
        </xsl:element>
        <xsl:value-of select="//tei:material" />
    </xsl:element>
    <xsl:element name="p"
     namespace="http://www.tei-c.org/ns/1.0">
      <xsl:element name="fw"
       namespace="http://www.tei-c.org/ns/1.0">
        <xsl:text>Numero di carte</xsl:text>
      </xsl:element>
      <xsl:value-of
       select="//tei:extent/tei:measure"/>
    </xsl:element>
```

```
<xsl:element name="p"
 namespace="http://www.tei-c.org/ns/1.0">
  <xsl:element name="fw"
   namespace="http://www.tei-c.org/ns/1.0">
    <xsl:text>Dimensioni </xsl:text>
  </xsl:element>
  <xsl:value-of
   select="//tei:height"/><xsl:text>x</xsl:te
   xt>
  <xsl:value-of select="//tei:width"/>
  <xsl:element name="unit"
     namespace="http://www.tei-
     c.org/ns/1.0">mm</xsl:element>
</xsl:element>
<xsl:element name="p"
     namespace="http://www.tei-c.org/ns/1.0">
  <xsl:element name="fw"
     namespace="http://www.tei-c.org/ns/1.0">
      <xsl:text>Watermark</xsl:text>
  </xsl:element>
  <xsl:variable name="watermark"
   select="//tei:watermark"/>
    <xsl:if test="exists($watermark)">
     <xsl:value-of select="$watermark" />
    </xsl:if>
    <xsl:if test="empty($watermark)">
      Non presente
    </xsl:if>
</xsl:element>
<xsl:element name="p"
 namespace="http://www.tei-c.org/ns/1.0">
    <xsl:element name="fw"
      namespace="http://www.tei-c.org/ns/1.0">
       <xsl:text>Timbri</xsl:text>
    </xsl:element>
    <xsl:value-of
      select="//tei:support/tei:stamp" />
```

```
        </xsl:element>
        <xsl:element name="p"
         namespace="http://www.tei-c.org/ns/1.0">
          <xsl:value-of select="//tei:support/tei:p"
           />
        </xsl:element>
      </xsl:element>
      <xsl:element name="div"
       namespace="http://www.tei-c.org/ns/1.0">
        <xsl:attribute name="type">piegatura
        </xsl:attribute>
        <xsl:element name="p"
         namespace="http://www.tei-c.org/ns/1.0">
        <xsl:element name="fw"
         namespace="http://www.tei-c.org/ns/1.0">
          Descrizione della piegatura
        </xsl:element>
        <xsl:value-of select="//tei:collation"/>
      </xsl:element>
    </xsl:element>
    <xsl:element name="div" namespace="http://www.tei-
     c.org/ns/1.0">
      <xsl:attribute name="type">condFisica
      </xsl:attribute>
      <xsl:element name="p" namespace="http://www.tei-
       c.org/ns/1.0">
        <xsl:element name="fw"
         namespace="http://www.tei-c.org/ns/1.0">
          Condizioni fisiche
        </xsl:element>
        <xsl:value-of select="//tei:condition"/>
      </xsl:element>
  </xsl:element>
  <xsl:element name="div" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:attribute name="type">handnote
    </xsl:attribute>
```

```
  <xsl:element name="p" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:element name="fw"
     namespace="http://www.tei-c.org/ns/1.0">
      Descrizione delle mani
    </xsl:element>
    <xsl:value-of select="//tei:handNote"/>
  </xsl:element>
</xsl:element>
<xsl:element name="div" namespace="http://www.tei-
 c.org/ns/1.0">
  <xsl:attribute name="type">sigilli
  </xsl:attribute>
  <xsl:element name="p" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:element name="fw"
     namespace="http://www.tei-c.org/ns/1.0">
      Descrizione dei sigilli
    </xsl:element>
    <xsl:variable name="sigilli" select="normalize-
     space(//tei:sealDesc)"/>
    <xsl:if test="exists($sigilli)">
      <xsl:value-of select="$sigilli" />
    </xsl:if>
    <xsl:if test="empty($sigilli)">Non presenti
    </xsl:if>
  </xsl:element>
</xsl:element>
<xsl:element name="div" namespace="http://www.tei-
 c.org/ns/1.0">
  <xsl:attribute name="type">infoAgg
  </xsl:attribute>
  <xsl:element name="p" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:element name="fw"
     namespace="http://www.tei-c.org/ns/1.0">
      Informazioni aggiuntive
```

```
  </xsl:element>
  <xsl:value-of select="normalize-
   space(//tei:additional)"/>
    </xsl:element>
</xsl:element>
<xsl:element name="div" namespace="http://www.tei-
 c.org/ns/1.0">
  <xsl:attribute name="type">Corrispondence
  </xsl:attribute>
  <xsl:element name="head"
    namespace="http://www.tei-c.org/ns/1.0">
    Informazioni sulla Corrispondenza
  </xsl:element>
  <xsl:element name="p" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:element name="fw"
     namespace="http://www.tei-c.org/ns/1.0">
      <xsl:text>Mittente</xsl:text>
    </xsl:element>
    <xsl:value-of
     select="//tei:correspAction[@type='sent']/tei
     :persName"/>
  </xsl:element>
  <xsl:element name="p" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:element name="fw"
       namespace="http://www.tei-c.org/ns/1.0">
      <xsl:text>Luogo di spedizione</xsl:text>
    </xsl:element>
    <xsl:variable name="luogosped"
      select="//tei:correspAction[@type='sent']/t
      ei:placeName"/>
     <xsl:if test="exists($luogosped)">
       <xsl:if test="$luogosped!='unknown'">
         <xsl:value-of select="$luogosped"/>
       </xsl:if>
     </xsl:if>
```

```
    <xsl:if test="empty($luogosped)">
    Non presente</xsl:if>
    <xsl:if test="$luogosped='unknown'">
    Non presente</xsl:if>
    <xsl:if test="$luogosped='sconosciuta'">
    Non presente</xsl:if>
    <xsl:if test="$luogosped='sconosciuto'">
    Non presente</xsl:if>
</xsl:element>
<xsl:element name="p" namespace="http://www.tei-
 c.org/ns/1.0">
  <xsl:element name="fw"
   namespace="http://www.tei-c.org/ns/1.0">
    <xsl:text>Data di spedizione</xsl:text>
  </xsl:element>
  <xsl:variable name="datasped"
   select="//tei:correspAction[@type='sent']/tei
   :date"/>
  <xsl:if test="exists($datasped)">
    <xsl:if test="$datasped!='unknown'">
      <xsl:value-of select="$datasped"/>
    </xsl:if>
  </xsl:if>
  <xsl:if test="empty($datasped)">
  Non presente</xsl:if>
  <xsl:if test="$datasped='unknown'">
  Non presente</xsl:if>
  <xsl:if test="$datasped='sconosciuta'">
  Non presente</xsl:if>
  <xsl:if test="$datasped='sconosciuto'">
  Non presente</xsl:if>
</xsl:element>
<xsl:element name="p" namespace="http://www.tei-
 c.org/ns/1.0">
  <xsl:element name="fw"
   namespace="http://www.tei-c.org/ns/1.0">
    <xsl:text>Destinatario</xsl:text>
```

```
</xsl:element>
<xsl:variable name="destinatario"
 select="//tei:correspAction[@type='receiver']
 /tei:persName"/>
<xsl:if test="exists($destinatario)">
  <xsl:if test="$destinatario!='unknown'">
     <xsl:value-of
      select="$destinatario"/></xsl:if>
  </xsl:if>
  <xsl:if test="empty($destinatario)">
  Non presente</xsl:if>
  <xsl:if test="$destinatario='unknown'">
  Non presente</xsl:if>
  <xsl:if test="$destinatario='sconosciuta'">
  Non presente</xsl:if>
  <xsl:if test="$destinatario='sconosciuto'">
  Non presente</xsl:if>
</xsl:element>
<xsl:element name="p"
 namespace="http://www.tei-c.org/ns/1.0">
  <xsl:element name="fw"
   namespace="http://www.tei-c.org/ns/1.0">
    <xsl:text>
    Luogo di destinazione</xsl:text>
  </xsl:element>
  <xsl:variable name="luogodest"
   select="//tei:correspAction[@type='receive
   r']/tei:placeName"/>
    <xsl:if test="exists($luogodest)">
      <xsl:if test="$luogodest!='unknown'">
        <xsl:value-of select="$luogodest"/>
      </xsl:if>
    </xsl:if>
    <xsl:if test="empty($luogodest)">
    Non presente</xsl:if>
    <xsl:if test="$luogodest='unknown'">
    Non presente</xsl:if>
```

```
                    <xsl:if test="$luogodest='sconosciuta'">
                     Non presente</xsl:if>
                    <xsl:if test="$luogodest='sconosciuto'">
                     Non presente</xsl:if>
                   </xsl:element>
                   <xsl:element name="p"
                    namespace="http://www.tei-c.org/ns/1.0">
                     <xsl:element name="fw"
                      namespace="http://www.tei-c.org/ns/1.0">
                       <xsl:text>Data di destinazione
                       </xsl:text>
                   </xsl:element>
                     <xsl:variable name="datasped"
                      select="//tei:correspAction[@type='recei
                      ver']/tei:date"/>
                     <xsl:if test="exists($datasped)">
                       <xsl:if test="$datasped!='unknown'">
                         <xsl:value-of select="$datasped"/>
                       </xsl:if>
                     </xsl:if>
                     <xsl:if test="empty($datasped)">
                     Non presente</xsl:if>
                     <xsl:if test="$datasped='unknown'">
                     Non presente</xsl:if>
                     <xsl:if test="$datasped='sconosciuta'">
                     Non presente</xsl:if>
                     <xsl:if test="$datasped='sconosciuto'">
                     Non presente</xsl:if>
               </xsl:element>
        </xsl:element>
    </xsl:element>
</xsl:template>
```

Di seguito il codice del template implementato per l'elemento `pb`.

```
<xsl:template match="tei:pb">
    <xsl:variable name="idno"
    select="//tei:idno[@type='inventory']"/>
    <xsl:variable name="id" select="@xml:id"/>
    <xsl:variable name="n" select="@n"/>
    <xsl:variable name="facs" select="@facs"/>
    <xsl:variable name="newN" select="concat ($idno, '.',
    $n)"/>
    <xsl:variable name="newidno" select="translate ($idno,
    '.', '-' )"/>
    <xsl:variable name="newfacs" select="concat ('data/test-
   img/', $idno, '/', $newidno, '_000', $n, '.dzi')"/>
    <xsl:element name="pb" namespace="http://www.tei-
    c.org/ns/1.0">
      <xsl:attribute name="n">
        <xsl:copy-of select="$newN"/>
      </xsl:attribute>
      <xsl:attribute name="xml:id">
        <xsl:copy-of select="$id"/>
      </xsl:attribute>
      <xsl:attribute name="facs">
        <xsl:copy-of select="$newfacs"/>
      </xsl:attribute>
      </xsl:element>
        <xsl:for-each select="//tei:surface[@xml:id=substring-
         after($facs,'#')]">
          <xsl:choose>
            <xsl:when
            test="//tei:idno[@type='inventory']='LL1.34.I' or
            //tei:idno[@type='inventory']='LL1.14' or
            //tei:idno[@type='inventory']='LL1.13.II'">
              <xsl:choose>
```

```
      <xsl:when
      test="exists(./tei:zone/tei:zone[@rendition='L
      ine'])">
      </xsl:when>
      <xsl:when
      test="exists(./tei:zone[@rendition='Line'])">
      </xsl:when>
      <xsl:otherwise>
        <xsl:element name="div"
        namespace="http://www.tei-c.org/ns/1.0">
          <xsl:attribute name="type">extratext
          </xsl:attribute>
          <xsl:element name="p"
           namespace="http://www.tei-c.org/ns/1.0">
          La facciata non presenta testo autografo
           di Bellini relativo a questa lettera
           </xsl:element>
        </xsl:element>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:when>
  <xsl:otherwise>
    <xsl:choose>
      <xsl:when
      test="exists(./tei:zone[@rendition='Line'])"><
      /xsl:when>
      <xsl:otherwise>
        <xsl:element name="div"
        namespace="http://www.tei-c.org/ns/1.0">
          <xsl:attribute name="type">extratext
          </xsl:attribute>
          <xsl:element name="p"
           namespace="http://www.tei-c.org/ns/1.0">
          La facciata non presenta testo autografo
         di Bellini relativo a questa lettera
         </xsl:element>
        </xsl:element>
```

```
            </xsl:otherwise>
          </xsl:choose>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:for-each>
  </xsl:template>
```

Di seguito il codice del template implementato per l'elemento `rs`.

```
<xsl:template match="tei:rs">
    <xsl:variable name="ref" select="@ref"/>
    <xsl:variable name="type" select="@type"/>
    <xsl:variable name="role" select="@role"/>
    <xsl:choose>
      <xsl:when test="$type='person'">
        <xsl:element name="persName"
         namespace="http://www.tei-c.org/ns/1.0">
          <xsl:if test="exists($ref)">
            <xsl:attribute name="ref">
              <xsl:value-of select="substring-after($ref,
              'TEI-ListPerson.xml')"/>
            </xsl:attribute>
          </xsl:if>
          <xsl:if test="exists($type)">
            <xsl:attribute name="type">
              <xsl:value-of select="$type"/>
            </xsl:attribute>
          </xsl:if>
          <xsl:if test="exists($role)">
            <xsl:attribute name="role">
              <xsl:value-of select="$role"/>
            </xsl:attribute>
          </xsl:if>
          <xsl:apply-templates/>
        </xsl:element>
      </xsl:when>
      <xsl:otherwise>
        <xsl:copy>
```

```
          <xsl:if test="exists($ref)">
            <xsl:attribute name="ref">
              <xsl:value-of select="substring-after($ref,
              'TEI-ListPerson.xml')"/>
            </xsl:attribute>
          </xsl:if>
          <xsl:if test="exists($type)">
            <xsl:attribute name="type">
              <xsl:value-of select="$type"/>
            </xsl:attribute>
          </xsl:if>
          <xsl:if test="exists($role)">
            <xsl:attribute name="role">
              <xsl:value-of select="$role"/>
            </xsl:attribute>
          </xsl:if>
          <xsl:apply-templates/>
        </xsl:copy>
      </xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

Di seguito il codice del template implementato per l'elemento `rs[@type='work']`.

```
<xsl:template match="tei:text//tei:rs[@type='work']">
     <xsl:copy>
       <xsl:apply-templates select="@*|node()"/>
     </xsl:copy>
     <xsl:element name="note" namespace="http://www.tei-
     c.org/ns/1.0">
       <xsl:attribute name="type">comment</xsl:attribute>
       <xsl:attribute name="n">0</xsl:attribute>
       <xsl:variable name="ref" select="@ref"/>
       <xsl:variable name="new" select="substring-after($ref,
        'TEI-ListWork.xml#')"/>
```

```
<xsl:variable name="listWork" select="doc('lists/TEI-
 ListWork.xml')//div//bibl/@xml:id"/>
<xsl:if test="$listWork">
  <xsl:element name="hi" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:attribute name="rend">bold</xsl:attribute>
    <xsl:text >Titolo: </xsl:text>
  </xsl:element>
  <xsl:value-of select="doc('lists/TEI-
   ListWork.xml')//div//bibl[@xml:id=$new]/title"/>
  <xsl:element name="hi" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:attribute name="rend">bold</xsl:attribute>
    <xsl:text >Compositore: </xsl:text>
  </xsl:element>
  <xsl:value-of select="doc('lists/TEI-
   ListWork.xml')//div//bibl[@xml:id=$new]//mei:compose
   r"/>
  <xsl:element name="hi" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:attribute name="rend">bold</xsl:attribute>
    <xsl:text >Librettista: </xsl:text>
  </xsl:element>
  <xsl:value-of select="doc('lists/TEI-
   ListWork.xml')//div//bibl[@xml:id=$new]/mei:libretti
   st"/>
  <xsl:element name="hi" namespace="http://www.tei-
   c.org/ns/1.0">
    <xsl:attribute name="rend">bold</xsl:attribute>
    <xsl:text >Prima rappresentazione: </xsl:text>
  </xsl:element>
  <xsl:value-of select="doc('lists/TEI-
   ListWork.xml')//div//bibl[@xml:id=$new]/orgName"/>
  <xsl:text >, </xsl:text>
  <xsl:value-of select="doc('lists/TEI-
   ListWork.xml')//div//bibl[@xml:id=$new]/placeName"/>
  <xsl:text >, </xsl:text>
```

```
          <xsl:value-of select="doc('lists/TEI-
           ListWork.xml')//div//bibl[@xml:id=$new]/date"/>
          <xsl:element name="hi" namespace="http://www.tei-
           c.org/ns/1.0">
             <xsl:attribute name="rend">bold</xsl:attribute>
             <xsl:text >Note: </xsl:text>
           </xsl:element>
           <xsl:value-of select="doc('lists/TEI-
            ListWork.xml')//div//bibl[@xml:id=$new]/note"/>
       </xsl:if>
    </xsl:element>
</xsl:template>
```

Di seguito il codice del template implementato per gli elementi `persName`, `placeName` e `orgName` .

```
<xsl:template match="tei:persName">
    <xsl:copy>
      <xsl:variable name="ref" select="@ref"/>
      <xsl:if test="exists($ref)">
        <xsl:attribute name="ref">
          <xsl:value-of select="substring-after($ref, 'TEI-
           ListPerson.xml')"/>
        </xsl:attribute>
      </xsl:if>
      <xsl:apply-templates/>
    </xsl:copy>
</xsl:template>


<xsl:template match="tei:placeName">
    <xsl:copy>
      <xsl:variable name="ref" select="@ref"/>
      <xsl:if test="exists($ref)">
        <xsl:attribute name="ref">
```

```
            <xsl:value-of select="substring-after($ref, 'TEI-
            ListPlace.xml')"/>
          </xsl:attribute>
        </xsl:if>
        <xsl:apply-templates/>
      </xsl:copy>
</xsl:template>


<xsl:template match="tei:orgName">
    <xsl:copy>
      <xsl:variable name="ref" select="@ref"/>
      <xsl:if test="exists($ref)">
        <xsl:attribute name="ref">
          <xsl:value-of select="substring-after($ref, 'TEI-
           ListOrganization.xml')"/>
        </xsl:attribute>
      </xsl:if>
      <xsl:apply-templates/>
    </xsl:copy>
</xsl:template>
```

Di seguito il codice del template implementato per l'elemento `back`.

```
<xsl:template match="tei:back">
    <xsl:element name="back" namespace="http://www.tei-
    c.org/ns/1.0">
      <xsl:element name="div" namespace="http://www.tei-
      c.org/ns/1.0">
         <xsl:attribute name="type">hotspot</xsl:attribute>
         <xsl:variable name="facs"
          select="//tei:teiHeader//*[@facs]"/>
         <xsl:variable name="zone"
          select="//tei:zone[@rendition='HotSpot']"/>
           <xsl:for-each select="$zone">
             <xsl:variable name="zone2" select="@xml:id"/>
```

```
      <xsl:variable name="newid" select="concat
      ($zone2, 'h')"/>
      <xsl:variable name="newfacs" select="concat('#',
      $zone2)"/>
      <xsl:element name="div"
       namespace="http://www.tei-c.org/ns/1.0">
        <xsl:attribute name="xml:id">
          <xsl:value-of select="$newid"/>
        </xsl:attribute>
        <xsl:attribute name="facs">
          <xsl:value-of select="$newfacs"/>
        </xsl:attribute>
        <xsl:for-each select="$facs">
          <xsl:variable name="facs3" select="@facs"/>
          <xsl:variable name="facs4"
          select="translate($facs3, '#', '')"/>
          <xsl:if test="$facs4=$zone2">
            <xsl:element name="p"
             namespace="http://www.tei-c.org/ns/1.0">
              <xsl:value-of>
                <xsl:apply-templates mode="hotspot"/>
              </xsl:value-of>
            </xsl:element>
          </xsl:if>
        </xsl:for-each>
      </xsl:element>
    </xsl:for-each>
  </xsl:element>
  <xsl:if
   test="//tei:idno[@type='inventory']='LL1.35'">
    <xsl:element name="div" namespace="http://www.tei-
     c.org/ns/1.0">
      <xsl:attribute name="type">biblCompleta
      </xsl:attribute>
      <xsl:element name="listBibl"
      namespace="http://www.tei-c.org/ns/1.0">
```

```
            <xsl:variable name="listbibl"
            select="doc('lists/TEI-
            ListBibl.xml')//tei:back//tei:div//tei:listBibl
            "/>
            <xsl:copy-of select="$listbibl"/>
          </xsl:element>
        </xsl:element>
      </xsl:if>
    </xsl:element>
</xsl:template>
```

13.2. Applicazione realizzata mediante l'utilizzo della tecnologia Saxon-JS

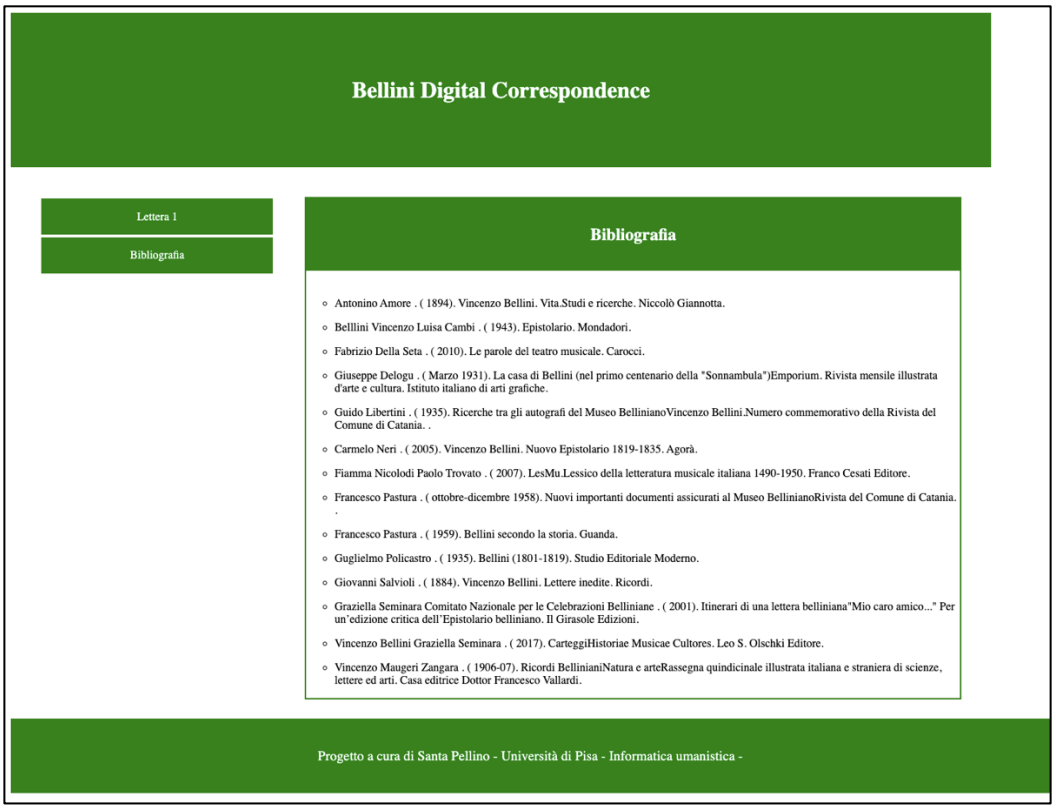

Figura 26. Interfaccia web della pagina *bibliografia.html* realizzata mediante l'uso della tecnologia Saxon-JS.

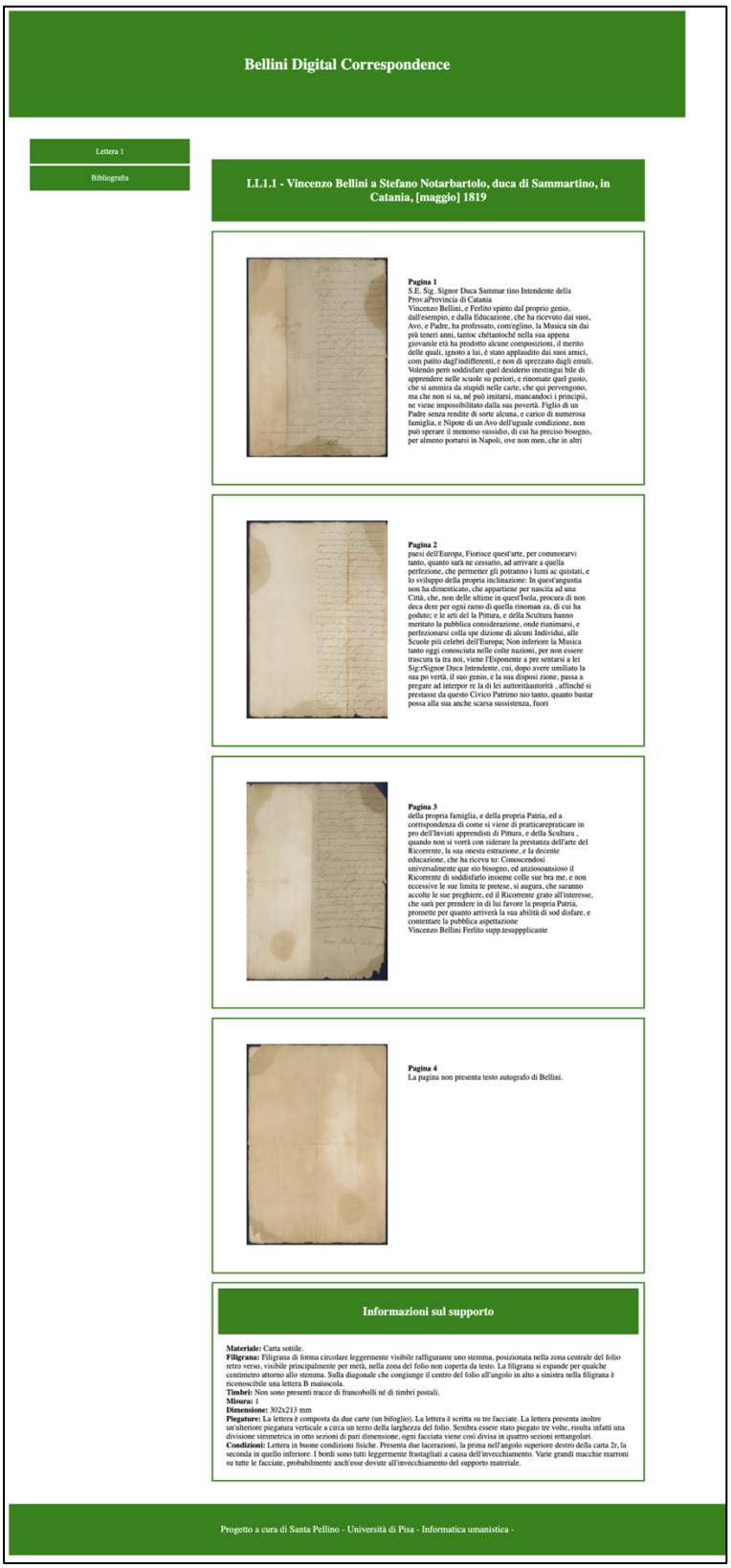

Figura 27. Interfaccia web della pagina *index.html* realizzata mediante l'uso della tecnologia Saxon-JS.

Documento *index.html*

(Percorso su GitHub Progetto-tesi-Santa-Pellino/Applicazione BDC Saxon/index.html).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Bellini Digital Correspondence</title>
    <meta HTTP-EQUIV="Content-Type" content="text/html;
    charset=UTF-8"/>
    <meta name="description" content="Bellini Digital
    Correspondence"/>
    <meta name="keywords" content="Vincenzo Bellini, Bellini
    Digital Correspondence, edizione digitale, lettere,
    corrispondenza"/>
    <meta name="author" content="Santa Pellino"/>
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0"/>
    <link rel="stylesheet" href="style.css"/>
    <script type="text/javascript" language="javascript"
    src="SaxonJS2.js"></script>
    <script>
    window.onload = function() {
    const
    sourcexml=["lettere/LL1_1.xml","lettere/LL1_2.xml"];
    SaxonJS.transform({
    stylesheetLocation: "trasforma.sef.json",
    sourceLocation: sourcexml[0]
    }, "async");}
    </script>
  </head>
  <body>
    <header>
      <h1 id="title"></h1> </header>
      <nav><ul>
        <li><a href="index.html">Lettera 1</a></li>
```

```
        <li><a href="bibliografia.html">Bibliografia</a></li>
        </ul>
      </nav>
      <div id="corpopagina" class="corpo">
        <div id="text"> </div>
        <div id="infosupporto" class="pagina"></div>
      </div>
    <footer>
       <div> Progetto a cura di Santa Pellino - Università di
        Pisa - Informatica umanistica -  </div>
    </footer>
  </body>
</html>
```

Documento *bellini.xsl*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/Applicazione BDC Saxon/bellini.xsl).

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:ixsl="http://saxonica.com/ns/interactiveXSLT"
xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-
prefixes="xs" extension-element-prefixes="ixsl" version="2.0"
xmlns:tei="http://www.tei-c.org/ns/1.0">
<xsl:template match="/">
  <xsl:result-document href="#title" method="ixsl:replace-
  content">
    Bellini Digital Correspondence
  </xsl:result-document>
  <xsl:result-document href="#text" method="ixsl:replace-
  content">
    <h2>
      <xsl:value-of select="//tei:idno[@type='inventory']"/>
      <xsl:value-of select="//tei:titleStmt//tei:title"/>
    </h2>
```

```
<xsl:for-each select="//tei:body//tei:pb">
 <xsl:choose>
   <xsl:when test="@n='1'">
     <div id="pagina1" class="pagina">
       <div id="image1" class="text-left">
         <xsl:variable name="link"
         select="concat('images/LL1-1_000',@n, '.jpg' )"/>
         <xsl:element name="img">
           <xsl:attribute name="src"><xsl:value-of
           select="$link"/>
           </xsl:attribute>
         </xsl:element>
       </div>
       <div id="text1" class="text-right" >
         <b>Pagina 1</b><br/>
          <xsl:copy-of
          select="//tei:body/tei:div[@xml:id='info_dest']"
          /><br />
         <xsl:copy-of
         select="//tei:body/tei:div/tei:ab[@n='ab_02']" />
       </div>
     </div>
  </xsl:when>
  <xsl:when test="@n='2'">
    <div id="pagina2" class="pagina">
      <div id="image2" class="text-left">
        <xsl:variable name="link"
         select="concat('images/LL1-1_000',@n, '.jpg' )"/>
        <xsl:element name="img">
          <xsl:attribute name="src">
            <xsl:value-of select="$link"/>
          </xsl:attribute>
        </xsl:element>
      </div>
      <div id="text2" class="text-right" >
        <b>Pagina 2</b><br/>
```

```
      <xsl:copy-of
      select="//tei:body/tei:div/tei:ab[@n='ab_03']"/>
    </div>
  </div>
</xsl:when>
<xsl:when test="@n='3'">
  <div id="pagina3" class="pagina">
    <div id="image3" class="text-left">
      <xsl:variable name="link"
       select="concat('images/LL1-1_000',@n, '.jpg' )"/>
      <xsl:element name="img"> <xsl:attribute
       name="src">
        <xsl:value-of select="$link"/></xsl:attribute>
      </xsl:element>
    </div>
    <div id="text3" class="text-right" >
      <b>Pagina 3</b><br/>
      <xsl:copy-of
      select="//tei:body/tei:div/tei:ab[@n='ab_04']"/>
      <xsl:copy-of
      select="//tei:body/tei:div/tei:salute[@n='ab_03']"
      /><br/>
      <xsl:copy-of select="//tei:signed[@hand='#h1']"/>
    </div>
  </div>
</xsl:when>
<xsl:when test="@n='4'">
  <div id="pagina4" class="pagina"><div id="image4"
   class="text-left">
    <xsl:variable name="link"
     select="concat('images/LL1-1_000',@n, '.jpg' )"/>
    <xsl:element name="img">
      <xsl:attribute name="src">
        <xsl:value-of select="$link"/>
      </xsl:attribute>
    </xsl:element>
  </div>
```

```
        <div id="text4" class="text-right" >
          <b>Pagina 4</b><br/>
           La pagina non presenta testo autografo di Bellini.
        </div>
        </div>
      </xsl:when>
      </xsl:choose>
    </xsl:for-each>
  <xsl:result-document href="#infosupporto"
  method="ixsl:replace-content">
    <h2>Informazioni sul supporto </h2>
    <br/><br/>
    <b>Materiale: </b>
    <xsl:value-of select="//tei:material"/>
    <br/>
    <b>Filigrana: </b>
    <xsl:value-of select="//tei:watermark"/><br/>
    <b>Timbri:
    </b><xsl:value-of select="//tei:stamp"/>
    <br/><b>Misura:
    </b><xsl:value-of select="//tei:measure"/><br/>
    <b>Dimensione: </b>
    <xsl:value-of select="//tei:dimensions/tei:height"/>
     x
    <xsl:value-of select="//tei:dimensions/tei:width"/> mm
     <br/>
    <b>Piegature: </b><xsl:value-of select="//tei:foliation"/>
    <br/><b>Condizioni:
    </b><xsl:value-of select="//tei:condition"/>
  </xsl:result-document>
 </xsl:template>
</xsl:transform>
```

Documento *bibliografia.html*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/Applicazione BDC Saxon/bibliografia.html).

```
<!DOCTYPE html>
<html>
  <head>
  <title>Bellini Digital Correspondence</title>
  <meta HTTP-EQUIV="Content-Type" content="text/html;
  charset=UTF-8"/>
  <meta name="description" content="Bellini Digital
  Correspondence"/>
  <meta name="keywords" content="Vincenzo Bellini, Bellini
  Digital Correspondence, edizione digitale, lettere,
  corrispondenza"/>
  <meta name="author" content="Santa Pellino"/>
  <meta name="viewport" content="width=device-width, initial-
  scale=1.0"/>
  <link rel="stylesheet" href="style.css"/>
  <script type="text/javascript" language="javascript"
  src="SaxonJS2.js"></script>
  <script>
    window.onload = function() {
      SaxonJS.transform({
      stylesheetLocation: "bibliografia.sef.json",
      sourceLocation: "lists/TEI-ListBibl.xml"
      }, "async");
    }
  </script>
  </head>
  <body>
    <header>
      <h1 id="title"></h1>
    </header>
      <nav>
        <ul>
```

```
          <li>
            <a href="index.html">Lettera 1</a>
          </li>
          <li>
            <a href="bibliografia.html">Bibliografia</a>
          </li>
        </ul>
      </nav>
      <div id="corpopagina" class="bibl">
        <div id="text" class="biblio"> </div>
      </div>
    <footer>
      <div> Progetto a cura di Santa Pellino - Università di
       Pisa - Informatica umanistica –
      </div>
    </footer>
  </body>
</html>
```

Documento bibliografia.xsl

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/Applicazione BDC Saxon/bibliografia.xsl)

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:ixsl="http://saxonica.com/ns/interactiveXSLT"
xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-
prefixes="xs" extension-element-prefixes="ixsl" version="2.0"
xmlns:tei="http://www.tei-c.org/ns/1.0"><xsl:template
match="/">
  <xsl:result-document href="#title" method="ixsl:replace-
  content">
  Bellini Digital Correspondence
  </xsl:result-document>
```

```
<xsl:result-document href="#text" method="ixsl:replace-
content">
  <h2>Bibliografia</h2>
  <xsl:for-each select="//tei:biblStruct">
    <ul>
     <li>
      <xsl:copy-of
       select=".//tei:author//text()|.//tei:editor//text()"/
       >. (
      <xsl:copy-of select=".//tei:date//text()"/>).
      <xsl:copy-of select=".//tei:title//text()"/>.
      <xsl:copy-of select=".//tei:publisher//text()"/>.
    </li>
    </ul>
  </xsl:for-each>
</xsl:result-document>
```

13.3. Applicazione creata con la tecnologia XQuery tramite l'utilizzo del processore Saxon

Figura 28. Interfaccia web della pagina *listapersone.html* dell'applicazione creata con XQuery tramite l'utilizzo del processore Saxon.

Figura 29. Interfaccia web della pagina *lettera.html* dell'applicazione creata con XQuery tramite l'utilizzo del processore Saxon.

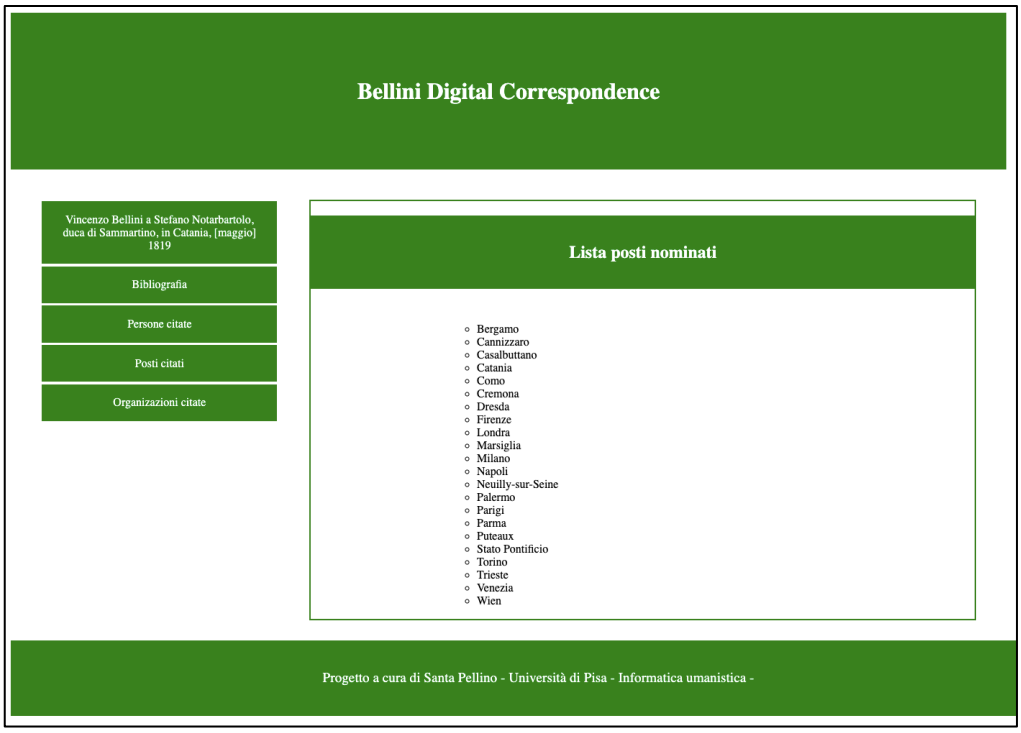

Figura 17. Interfaccia web della pagina *listaposti.html* dell'applicazione creata con XQuery tramite l'utilizzo del processore Saxon.

Figura 18. Interfaccia web della pagina *listaorg.html* dell'applicazione creata con XQuery tramite l'utilizzo del processore Saxon.

Figura 19. Interfaccia web della pagina *bibliografia.html* dell'applicazione creata con XQuery tramite l'utilizzo del processore Saxon.

Documento *lettera.xql*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/Applicazione BDC XQuery/lettera.xql).

```
xquery version "3.1";
  declare namespace tei = "http://www.tei-c.org/ns/1.0";
  declare namespace output = "http://www.w3.org/2010/xslt-xquery-
   serialization";
  declare option output:media-type "text/html";
  declare variable $page-title := "Bellini Digital Correspondence";
  <html>
    <meta HTTP-EQUIV="Content-Type" content="text/html; charset=UTF-
    8"/>
    <meta name="description" content="Bellini Digital
    Correspondence"/>
    <meta name="keywords" content="Vincenzo Bellini, Bellini Digital
    Correspondence, edizione digitale, lettere, corrispondenza"/>
    <meta name="author" content="Santa Pellino"/>
```

```
<meta name="viewport" content="width=device-width, initial-
 scale=1.0"/>
<link rel="stylesheet" href="style.css"/>
<body>
    <header>
         <h1> {$page-title}</h1>
    </header>
    <nav><ul>
      <li><a href="lettera.html">{
        for $title in
        doc("letter/LL1_1.xml")//tei:titleStmt/tei:title
        return $title/text()}</a></li>
      <li><a href="bibliografia.html">Bibliografia</a></li>
      <li><a href="listapersone.html">Persone citate</a></li>
      <li><a href="listaposti.html">Posti citati</a></li>
      <li><a href="listaorg.html">Organizzazioni citate</a></li>
    </ul></nav>
    <div id="corpopagina" class="corpo">{
      for $title in
       doc("letter/LL1_1.xml")//tei:titleStmt/tei:title
      return <h2>{$title/text()}</h2>}
    <div id="pagina1" class="pagina">
      <b>Immagini delle facciate della lettera:</b>
    <div>
      <img src="images/LL1-1_0001.jpg" width="300px" />
       <img src="images/LL1-1_0002.jpg" width="300px" />
      <img src="images/LL1-1_0003.jpg" width="300px" />
      <img src="images/LL1-1_0004.jpg" width="300px" />
    </div>
    <div id="text1" class="text-left" >
      <b>Testo della lettera:</b>
      <p>{for $testo in
       doc("letter/LL1_1.xml")//tei:text//tei:body
      return $testo//tei:div[@type='letter-body']//text()}</p>
    </div>
    <div id="info" class="text-left" >
      <b>Informazioni sul supporto</b>
      <p>{
      for $testo in doc("letter/LL1_1.xml")//tei:supportDesc
      return $testo//tei:support/tei:p//text()
        <b>Materiale: </b><br/>{
```

```
            for $testo in doc("letter/LL1_1.xml")//tei:supportDesc
            return $testo//tei:material//text() }<br/>
              <b>Filigrana: </b><br/>{
            for $testo in doc("letter/LL1_1.xml")//tei:supportDesc
            return $testo//tei:watermark//text() }<br/>
              <b>Timbri: </b><br/>{
            for $testo in doc("letter/LL1_1.xml")//tei:supportDesc
            return $testo//tei:stamp//text() }<br/>
              <b>Dimensioni: </b><br/>{
            for $testo in doc("letter/LL1_1.xml")//tei:supportDesc
            return $testo//tei:height//text() }x{for $testo in
             doc("letter/LL1_1.xml")//tei:supportDesc
            return $testo//tei:width//text() }mm <br/>
              <b>Piegature: </b><br/>{
            for $testo in doc("letter/LL1_1.xml")//tei:supportDesc
            return $testo//tei:collation//text() }<br/>
              <b>Condizioni fisiche: </b><br/>{
            for $testo in doc("letter/LL1_1.xml")//tei:supportDesc
            return $testo//tei:condition//text() }</p>
          </div>
          <div id="info-mani" class="text-left-mani" >
            <b>Altre mani</b><ul>{
            for $testo in
             doc("letter/LL1_1.xml")//tei:handDesc//tei:handNote
            return <li>{$testo//text() }</li>}</ul>
          </div>
        </div>
      </div>
      <footer>
        <div> Progetto a cura di Santa Pellino - Università di Pisa -
         Informatica umanistica –
        </div>
      </footer>
    </body>
</html>
```

Documento *listaorg.xql*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/Applicazione BDC XQuery/listaorg.xql).

```
xquery version "3.1";
declare namespace tei = "http://www.tei-c.org/ns/1.0";
declare namespace output = "http://www.w3.org/2010/xslt-
xquery-serialization";
declare option output:media-type "text/html";
declare variable $page-title := "Bellini Digital
Correspondence";
<html>
  <meta HTTP-EQUIV="Content-Type" content="text/html;
  charset=UTF-8"/ù
  <meta name="description" content="Bellini Digital
  Correspondence"/>
  <meta name="keywords" content="Vincenzo Bellini, Bellini
  Digital Correspondence, edizione digitale, lettere,
  corrispondenza"/>
  <meta name="author" content="Santa Pellino"/>
  <meta name="viewport" content="width=device-width, initial-
  scale=1.0"/>
  <link rel="stylesheet" href="style.css"/>
  <body>
    <header>
      <h1> {$page-title}</h1>
    </header>
    <nav><ul><li><a href="lettera.html">{
      for $title in
       doc("letter/LL1_1.xml")//tei:titleStmt/tei:title
      return $title/text()}</a></li>
      <li><a href="bibliografia.html">Bibliografia</a></li>
      <li><a href="listapersone.html">Persone citate</a></li>
      <li><a href="listaposti.html">Posti citati</a></li>
      <li><a href="listaorg.html">Organizzazioni
       citate</a></li>
      </ul></nav>
```

```
    <div id="corpopagina" class="lista">
      <h2>Lista organizzazioni nominate</h2>
      <div class="list"><ul>
        {for $testo in doc("lists/TEI-
         ListOrganization.xml")//tei:listOrg/tei:org/tei:orgNa
         me
        return <li>{$testo//text()}</li>}</ul>
      </div>
    </div>
    <footer>
      <div> Progetto a cura di Santa Pellino - Università di
       Pisa - Informatica umanistica –
      </div>
    </footer>
  </body>
</html>
```

Documento *listapersone.xql*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/Applicazione BDC XQuery/listapersone.xql).

```
xquery version "3.1";
declare namespace tei = "http://www.tei-c.org/ns/1.0";
declare namespace output = "http://www.w3.org/2010/xslt-
xquery-serialization";
declare option output:media-type "text/html";
declare variable $page-title := "Bellini Digital
Correspondence";
<html>
  <meta HTTP-EQUIV="Content-Type" content="text/html;
  charset=UTF-8"/>
  <meta name="description" content="Bellini Digital
  Correspondence"/>
  <meta name="keywords" content="Vincenzo Bellini, Bellini
  Digital Correspondence, edizione digitale, lettere,
  corrispondenza"/>
```

```
  <meta name="author" content="Santa Pellino"/>
  <meta name="viewport" content="width=device-width, initial-
  scale=1.0"/>
  <link rel="stylesheet" href="style.css"/>
  <body>
    <header>
      <h1> {$page-title}</h1>
    </header>
    <nav><ul><li><a href="lettera.html">{
      for $title in
       doc("letter/LL1_1.xml")//tei:titleStmt/tei:title
      return $title/text()}</a></li>
      <li><a href="bibliografia.html">Bibliografia</a></li>
      <li><a href="listapersone.html">Persone citate</a></li>
      <li><a href="listaposti.html">Posti citati</a></li>
      <li><a href="listaorg.html">Organizzazioni
       citate</a></li></ul>
    </nav>
    <div id="corpopagina" class="lista">
      <h2>Lista persone nominate</h2>
      <div class="list"><ul>{
        for $testo in doc("lists/TEI-
         ListPerson.xml")//tei:listPerson//tei:person/tei:pers
         Name
        return <li>{$testo//text() }</li>}</ul>
      </div>
    </div>
    <footer>
      <div> Progetto a cura di Santa Pellino - Università di
       Pisa - Informatica umanistica –
      </div>
    </footer>
  </body>
</html>
```

Documento *listaposti.xql*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/Applicazione BDC XQuery/listaposti.xql).

```
xquery version "3.1";
declare namespace tei = "http://www.tei-c.org/ns/1.0";
declare namespace output = "http://www.w3.org/2010/xslt-
xquery-serialization";
declare option output:media-type "text/html";
declare variable $page-title := "Bellini Digital
Correspondence";
<html>
  <meta HTTP-EQUIV="Content-Type" content="text/html;
  charset=UTF-8"/>
  <meta name="description" content="Bellini Digital
  Correspondence"/>
  <meta name="keywords" content="Vincenzo Bellini, Bellini
  Digital Correspondence, edizione digitale, lettere,
  corrispondenza"/>
  <meta name="author" content="Santa Pellino"/>
  <meta name="viewport" content="width=device-width, initial-
  scale=1.0"/>
  <link rel="stylesheet" href="style.css"/>
  <body>
    <header>
      <h1>{$page-title}</h1>
    </header>
    <nav>
      <ul>
        <li><a href="lettera.html">
          {for $title in
           doc("letter/LL1_1.xml")//tei:titleStmt/tei:title
          return $title/text()}</a></li>
        <li><a href="bibliografia.html">Bibliografia</a></li>
        <li><a href="listapersone.html">Persone citate</a>
        </li>
        <li><a href="listaposti.html">Posti citati</a></li>
```

```
        <li><a href="listaorg.html">Organizzazioni citate</a>
        </li>
      </ul>
    </nav>
    <div id="corpopagina" class="lista">
      <h2>Lista posti nominati</h2>
      <div class="list"><ul>{
        for $testo in doc("lists/TEI-
         ListPlace.xml")//tei:listPlace/tei:place/tei:placeNam
         e
        return <li>{$testo//text() }</li>}</ul>
      </div>
    </div>
    <footer>
      <div> Progetto a cura di Santa Pellino - Università di
       Pisa - Informatica umanistica -
      </div>
     </footer>
  </body>
</html>
```

Documento *bibliografia.xql*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/Applicazione BDC XQuery/bibliografia.xql).

```
xquery version "3.1";
declare namespace tei = "http://www.tei-c.org/ns/1.0";
declare namespace output = "http://www.w3.org/2010/xslt-
xquery-serialization";
declare option output:media-type "text/html";
declare variable $page-title := "Bellini Digital
Correspondence";
<html>
  <meta HTTP-EQUIV="Content-Type" content="text/html;
  charset=UTF-8"/>
  <meta name="description" content="Bellini Digital
  Correspondence"/>
  <meta name="keywords" content="Vincenzo Bellini, Bellini
  Digital Correspondence, edizione digitale, lettere,
  corrispondenza"/>
  <meta name="author" content="Santa Pellino"/>
  <meta name="viewport" content="width=device-width, initial-
  scale=1.0"/>
  <link rel="stylesheet" href="style.css"/>
  <body>
    <header>
      <h1>{$page-title}</h1>
    </header>
      <nav>
        <ul>
          <li><a href="lettera.html">{
           for $title in
           doc("letter/LL1_1.xml")//tei:titleStmt/tei:title
          return $title/text()}</a>
          </li>
          <li><a
           href="bibliografia.html">Bibliografia</a></li>
          <li><a href="listapersone.html">Persone citate</a>
```

```
            </li>
            <li><a href="listaposti.html">Posti citati</a></li>
            <li><a href="listaorg.html">Organizzazioni
             citate</a>
            </li>
          </ul>
        </nav>
        <div id="corpopagina" class="bibl">
          <h2>Biblibliografia</h2>
          <div class="biblio">
            <ul>{
              for $biblio in doc("lists/TEI-
              ListBibl.xml")//tei:biblStruct
              return <li>
              {$biblio//tei:author//text()|$biblio//tei:editor//
              text()}. ({$biblio//tei:date//text()}).
              {$biblio//tei:title//text()}.{$biblio//tei:publish
              er//text()}.</li> }
            </ul>
          </div>
        </div>
      <footer>
      <div> Progetto a cura di Santa Pellino - Università di
       Pisa - Informaticaumanistica -  </div>
      </footer>
    </body>
</html>
```

Comandi per generare i documenti HTML tramite XQuery

```
java -cp saxon-he-10.3.jar net.sf.saxon.Query -q:lettera.xql -
o:lettera.html

java -cp saxon-he-10.3.jar net.sf.saxon.Query -
q:bibliografia.xql -o:bibliografia.html

java -cp saxon-he-10.3.jar net.sf.saxon.Query -
q:listapersone.xql -o:listapersone.html

java -cp saxon-he-10.3.jar net.sf.saxon.Query -
q:listaposti.xql -o:listaposti.html

java -cp saxon-he-10.3.jar net.sf.saxon.Query -q:listaorg.xql
-o:listaorg.html
```

13.4. Applet creata per eXist-db

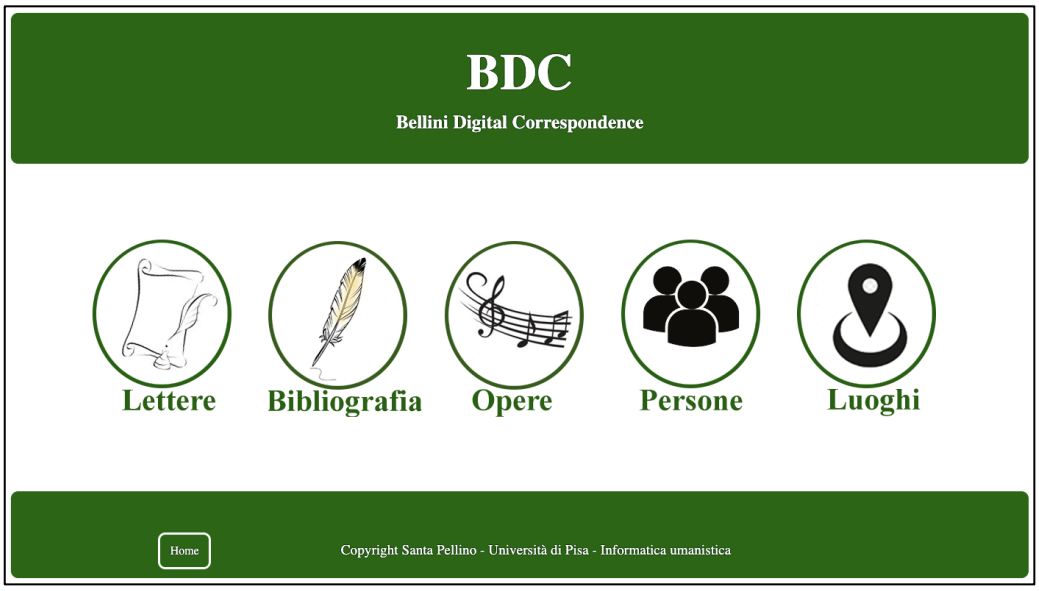

Figura 20. Interfaccia web della pagina *index.html* dell'applet cresta per eXist-db.

Figura 21. Interfaccia web della pagina *lettere.html* dell'applet cresta per eXist-db.

Figura 22. Interfaccia web della pagina *bibliografia.html* dell'applet cresta per eXist-db.

Figura 23. Interfaccia web della pagina *opere.html* dell'applet cresta per eXist-db.

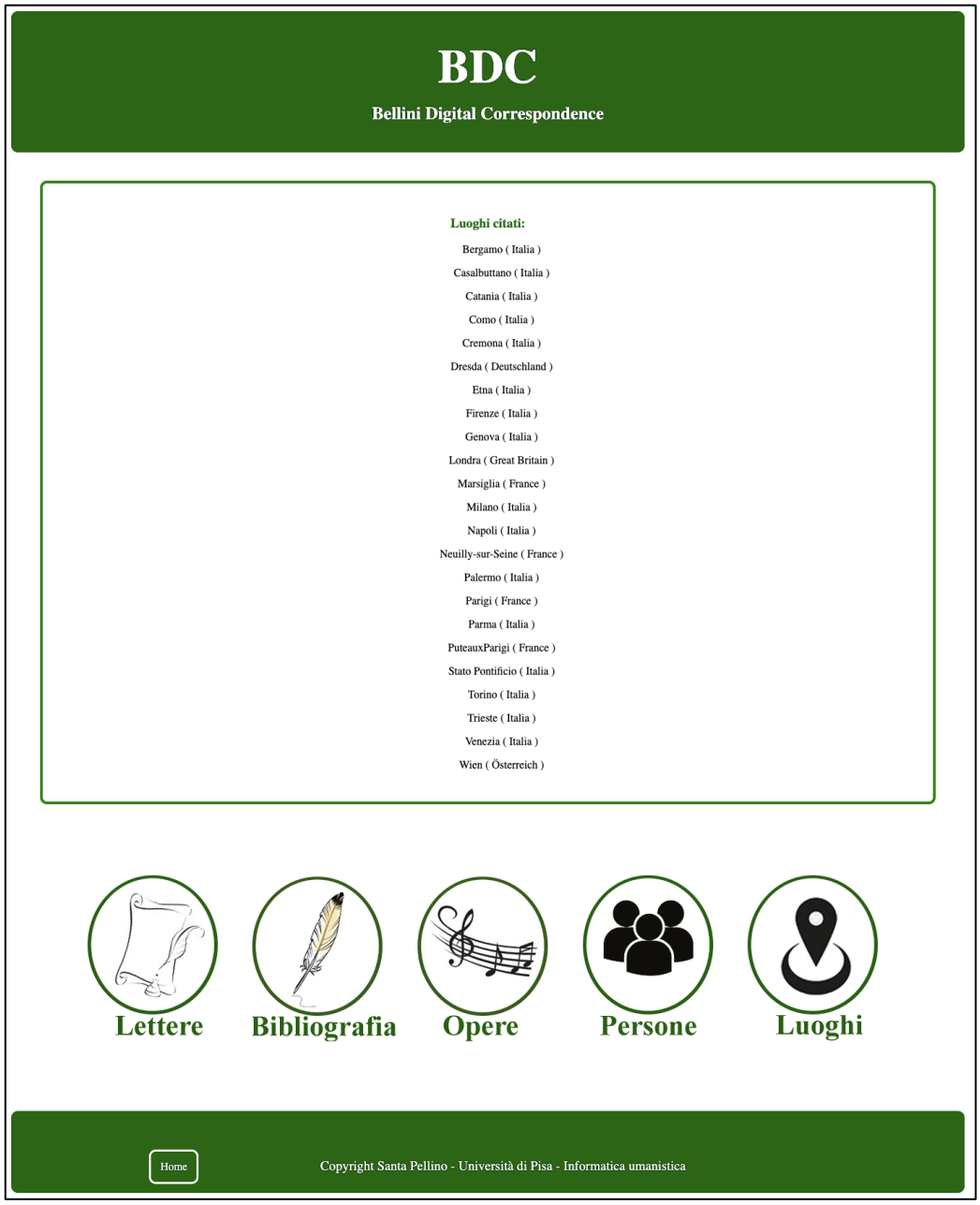

Figura 24. Interfaccia web della pagina *place.html* dell'applet cresta per eXist-db.

Figura 25. Interfaccia web della pagina *people.html* dell'applet cresta per eXist-db.

Documento *app.xql*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/ApplicazioneBDC eXist/file implementati/app.xql).

```
xquery version "3.1";
module namespace app="http://exist-db.org/yes/BDC/templates";
import module namespace templates="http://exist-
db.org/xquery/html-templating";
import module namespace lib="http://exist-db.org/xquery/html-
templating/lib";
import module namespace config="http://exist-
db.org/yes/BDC/config" at "config.xqm";
declare namespace tei = "http://www.tei-c.org/ns/1.0";
declare namespace mei="https://music-
encoding.org/schema/4.0.1/mei-all.rng";
declare %templates:wrap
function app:foo($node as node(), $model as map(*)) {
  <p>Dummy templating function.</p>};
  declare function app:test($node as node(), $model as map(*))
  {
    <p>Dummy template output generated by function app:test at
    {current-dateTime()}. The templating
        function was triggered by the class attribute
         <code>class="app:test"</code>.
    </p>};
  declare function app:biblio($node as node(), $model as
  map(*)) {
    for $biblio in doc("/db/apps/BDC/Lists/TEI-
    ListBibl.xml")//tei:biblStruct
    return <ul><li>Autore:
    {$biblio//tei:author//text()|$biblio//tei:editor//text()}
    . ({$biblio//tei:date//text()}).
    {$biblio//tei:title//text()}.
    {$biblio//tei:publisher//text()}.</li></ul>};
  declare function app:place($node as node(), $model as
  map(*)) {
```

```
    for $place in doc("/db/apps/BDC/Lists/TEI-
    ListPlace.xml")//tei:place
    return <ul><li><a href="{$place//tei:placeName/@ref}">
    {$place//tei:placeName//text()|$place//tei:geogName//text
    ()}({$place//tei:country//text()})</a></li></ul>};
  declare function app:people($node as node(), $model as
  map(*)) {
    for $people in doc("/db/apps/BDC/Lists/TEI-
    ListPerson.xml")//tei:person
    return <ul><li>{$people//tei:surname//text()}(
   {$people//tei:forename//text()})</li></ul};
  declare function app:opere($node as node(), $model as
  map(*)) {
    for $opere in doc("/db/apps/BDC/Lists/TEI-
    ListWork.xml")//listBibl/bibl
    return <ul><li>{$opere/title//text()}.
    Compositore: {$opere/mei:composer//text()}
    . Librettista: {$opere/mei:librettist//text()}
    . Luogo:{$opere/orgName//text()}
    di {$opere/placeName//text()}
    .Data:{$opere/date//text()}.</li></ul>};
  declare function app:lettere($node as node(), $model as
  map(*)) {
    for $lettere in collection("/db/apps/BDC/Lettere/")
    return <ul><li><a>{$lettere//tei:idno[@type='inventory']}
    - {$lettere//tei:titleStmt/tei:title/text()}</a></li></ul>
    };
```

Documento *BDCTemplate.html*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/ApplicazioneBDC eXist/file implementati/*BDCTemplate.html*).

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title data-template="config:app-title">BDC</title>
```

```
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0"/>
    <meta data-template="config:app-meta"/>
    <link rel="shortcut icon"
    href="resources/images/exist_icon_16x16.ico"/>
    <link rel="stylesheet" type="text/css"
    href="resources/css/style.css"/>
    <script type="text/javascript"
    src="resources/scripts/bootstrap.bundle.min.js"/>
  </head>
  <body>
    <div id="header">
      <h1>BDC</h1>
      <h2 data-template="config:app-title">Generated page</h2>
    </div>
    <div id="content"> Contenuto </div>
    <div id="footer">
      <a
      href="http://localhost:8080/exist/apps/BDC/index.html">
      Home</a>
      <p>Copyright Santa Pellino - Università di Pisa -
      Informatica umanistica</p>
    </div>
  </body>
</html>
```

Documento *bibliografia.html*

(Percorso su GitHub: Progetto-tesi-Santa-Pellino/ApplicazioneBDC eXist/file implementati/*bibliografia.html*).

```
<div xmlns="http://www.w3.org/1999/xhtml" data-
template="templates:surround" data-template-
with="templates/BDCTemplate.html" data-template-at="content">
  <div class="col-md-3">
    <div id="left">
      <h3>Bibliografia </h3>
      <div id="biblio" data-template="app:biblio"/>
```

```
    </div>
  </div>
  <div id="opzioni">
    <a
     href="http://localhost:8080/exist/apps/BDC/lettere.html
     ">
    <img src="resources/images/lettere.png"/></a>
    <a
     href="http://localhost:8080/exist/apps/BDC/bibliografia
     .html">
       <img src="resources/images/piumatesto.png"/></a>
     <a
      href="http://localhost:8080/exist/apps/BDC/opere.html"
      >
        <img src="resources/images/notatesto.png"/></a>
      <a
       href="http://localhost:8080/exist/apps/BDC/people.htm
       l">
         <img src="resources/images/peopletesto.png"/></a>
       <a
        href="http://localhost:8080/exist/apps/BDC/place.htm
        l">
          <img src="resources/images/luoghi.png"/></a>
  </div>
</div>
```