

Sistema di Visualizzazione TEI Roma P5

Progetto per il corso di Codifica di Testi

- **Studente:** Davide Caruso
- **Corso di Laurea:** Informatica Umanistica
- **Università:** Università di Pisa
- **Docente:** Prof. Angelo Mario del Grosso
- **Repository GitHub:** <https://github.com/davidecaruso03/codificaditesti>
- **Demo online:** <https://www.pisa.live/tei3>

1. Panoramica del Progetto

Questo progetto implementa un **sistema di visualizzazione interattiva per testi codificati in TEI Roma P5**. Il sistema permette di navigare simultaneamente i facsimili digitalizzati di un documento storico e la relativa trascrizione codificata, creando un collegamento visivo interattivo tra ogni riga di testo e la sua esatta posizione sull'immagine originale.

Caratteristiche Principali

- **Visualizzazione sincronizzata:** Collegamento interattivo tra immagini dei facsimili e testo codificato
- **Sistema multi-pagina dinamico:** Supporta automaticamente qualsiasi numero di pagine senza modifiche al codice (viene modificato solo il file testo.xml)
- **Zone interattive responsive:** Le coordinate delle zone si adattano automaticamente al ridimensionamento della finestra
- **Codifica semantica con 13 tipologie di fenomeni notevoli:** Persone, luoghi, navi, organizzazioni, date, titoli, citazioni, enfasi, esclamazioni, termini tecnici, ruoli, paesi e identificatori
- **Doppio sistema di interazione:** Zone visibili (overlay rosso) e zone invisibili per mantenere l'interattività

Obiettivo di Riusabilità

Un obiettivo chiave del progetto era **rendere la componente di visualizzazione indipendente dal contenuto specifico del file testo.xml**.

L'architettura è stata progettata in modo che il sistema possa essere impiegato per la codifica e la visualizzazione di testi completamente diversi senza la necessità di modificare il file di trasformazione (`.xsl`).

Standard e Tecnologie

- **Standard:** TEI Roma P5 versione 4.10.2
 - **Trasformazione:** XSLT 1.0
 - **Interattività:** JavaScript ES5 (compatibilità universale)
 - **Presentazione:** CSS3 con Flexbox
 - **Validazione:** DTD TEI completo (`tei_all.dtd`)
-

2. Flusso di Lavoro

Ho diviso lo sviluppo in 2 fasi:

- **Fase 1: Generazione dell'HTML direttamente dal browser a partire da xml + xsl**

Questo per 2 motivi:

1. Velocizzare la fase di sviluppo del codice, consentendo di modificare e testare il funzionamento del SW interattivamente e in tempo reale (una tantum).
2. Accelerare e semplificare la fase di messa a punto di una codifica che può essere verificata interattivamente mentre la si crea (ogni volta che si fa una codifica di un nuovo testo).

Quindi la Fase 1 è stata fatta inizialmente per creare la soluzione SW di trasformazione (file tei_transform.xsl) e viene ripetuta ogni volta che deve essere fatta la codifica digitale di un novo testo

- **Fase 2: Trasformazione finale con Saxon da xml + xsl a HTML**

Viene utilizzato l'HTML generato da Saxon per la versione conclusiva del progetto.

In questa fase il file tei_transform.xsl contiene al suo interno anche le componenti .css e .js che in Fase 1 erano in file separati e inclusi.

Il flusso di lavoro seguito nella **Fase 1** è:

- Estrazione delle Immagini:** Ho estratto le immagini del testo da codificare da un documento PDF (<https://rassegnavasettimanale.animi.it/wp-content/uploads/2019/02/Vol6-1880-2-S2-F144.pdf>).
- Trascrizione e Suddivisione:** Dalle immagini ho trascritto il testo, che ho poi suddiviso in righe e inserito nel foglio di calcolo `zone.xlsx`.
- Mappatura delle Zone:** Ho creato una mappatura tra le aree grafiche delle immagini e il testo corrispondente. Per far questo, ho utilizzato MS Paint per rilevare le coordinate degli angoli del rettangolo contenente ogni riga di testo e il foglio `zone.xlsx` per registrarle.

Struttura del Foglio Excel `zone.xlsx`

Per ogni sezione/pagina da codificare, è stata creata una pagina dedicata nel foglio Excel con una **struttura tabellare standardizzata** composta dalle seguenti colonne:

Colonna	Descrizione	Esempio
<code>id</code>	Identificatore univoco della zona	<code>ZONE_ZN_P1_R0</code>
<code>ulx</code>	Coordinata X angolo superiore sinistro	588
<code>uly</code>	Coordinata Y angolo superiore sinistro	408
<code>lrx</code>	Coordinata X angolo inferiore destro	1108
<code>lry</code>	Coordinata Y angolo inferiore destro	432
<code>Testo</code>	Contenuto testuale della riga	CORRISPONDENZA DA CASTELLAMMARE.

Esempio di righe dalla pagina 1:

<code>id</code>	<code>ulx</code>	<code>uly</code>	<code>lrx</code>	<code>lry</code>	Testo
<code>ZONE_ZN_P1_R0</code>	588	408	1108	432	CORRISPONDENZA DA CASTELLAMMARE.
<code>ZONE_ZN_P1_R1</code>	588	434	1108	444	IL VARO DELLA «Italia»
<code>ZONE_ZN_P1_R2</code>	588	446	1108	462	30 Settembre 1880.
<code>ZONE_ZN_P1_R3</code>	588	464	1108	488	Il varo dell'Italia è un fatto compiuto. Delle feste, degli

Questa struttura permette di:

- **Mantenere sincronizzati** ID, coordinate e testo in un'unica vista

- **Facilitare la verifica** della corrispondenza tra zone e contenuto
- **Automatizzare la generazione** del codice XML tramite macro VBA
- **Organizzare il lavoro** per pagina, mantenendo separati i diversi facsimili

4. **Generazione del Codice XML:** Tramite macro VBA nel foglio Excel (Sviluppo > Macro), ho generato automaticamente le sezioni di codice da inserire nel file `testo.xml`. Nello specifico:
 - La macro `EsportaTesti` ha creato i tag `<1>` contenenti le righe di testo.
 - La macro `EsportaZone` ha creato i tag `<zone>` con le relative coordinate.
 5. **Compilazione del teiHeader:** Nella sezione `<teiHeader>` del file `testo.xml` ho inserito tutte le informazioni dei metadati relativi al documento.
 6. **Codifica Semantica:** Ho individuato e codificato i "fenomeni notevoli" (nomi di persona, luoghi, date, etc.) presenti nel testo tramite gli appositi tag TEI.
 7. **Creazione dei File Web:** Ho creato i file `tei_transform.xsl`, `tei_transform.js` e `tei_transform.css`. Il file `tei_transform.xsl` (XSLT - Extensible Stylesheet Language Transformations) ha lo scopo fondamentale di trasformare il documento `testo.xml` in una pagina HTML, che può essere visualizzata da un browser. Legge i dati strutturati nell'XML e li riorganizza in elementi HTML, creando la struttura a due colonne e iniettando i dati necessari (come le coordinate delle zone) affinché gli script possano renderla interattiva.
-

3. Strumenti Software Utilizzati

Per la realizzazione del progetto, nella **Fase 1**, ho impiegato i seguenti strumenti:

- **Visual Studio Code:** Utilizzato come editor principale per tutti i file di codice, sfruttando le estensioni:
 - **XML** (di Red Hat): per la validazione della sintassi del file `testo.xml` (tramite `Ctrl+Shift+M`) utilizzando le definizioni presenti in `tei_all.dtd`.
 - **Markdown All in One** (di Yu Zhang): per la visualizzazione dell'anteprima della documentazione (`Ctrl+Shift+V`) scritta con sintassi Markdown.
- **MS Paint:** Impiegato per l'individuazione manuale delle coordinate in pixel delle zone di testo sulle immagini dei facsimili.
- **MS Excel:** Usato per censire il testo e le relative coordinate delle zone, e per generare automaticamente il codice XML tramite macro VBA.
- **LLM (Gemini e Claude):** Utilizzati come assistenti per i processi di verifica dei risultati, per la messa a punto delle componenti JavaScript e CSS, e per l'impaginazione della documentazione in formato Markdown. Gli LLM si sono rivelati particolarmente utili per accelerare lo sviluppo del frontend, aiutando a individuare colori con un contrasto adeguato, suggerendo le regole CSS corrette per risolvere problemi di allineamento e spiegando perché alcune parti del codice JavaScript non funzionassero come previsto (ad es. il refresh degli elementi in seguito al `resize` della finestra), suggerendo le relative correzioni.

Nella **Fase 2**, ho impiegato i seguenti strumenti:

- La versione Java di **Xerces 2.12.2** per la validazione finale dell'xml rispetto al dtd.
- La versione Java di **saxon-he-12.9** per la generazione dell'HTML da utilizzare

Nella directory xerces vi è il batch **EseguiXerces.bat** utilizzato per la validazione assieme ai file .xml e .dtd. La validazione con Xerces non ha prodotto errori.

Nella directory saxon vi è il batch **EseguiSaxon.bat** utilizzato per la trasformazione da xsl ad HTML. La trasformazione con Saxon non ha prodotto errori.

4. Architettura del Sistema (Fase 1)

Componenti Principali

- `testo.xml` : Documento TEI P5 contenente i metadati e il testo codificato della rivista "La Rassegna Settimanale".
- `tei_transform.xsl` : documento XSLT 1.0 per la trasformazione da XML a HTML.
- `tei_transform.js` : Script JavaScript (ES5) che gestisce la logica di interattività.
- `tei_transform.css` : Foglio di stile CSS3 per la formattazione e il layout.
- `tei_all.dtd` : Definizione DTD per la validazione dello schema TEI.
- `index.html` : Contenitore HTML che carica il documento trasformato (`testo.xml`) in un `<iframe>` (Nella Fase 2, questo file viene sostituito dall'output diretto di Saxon).
- `Immagini JPG` : 6 facsimili digitalizzati delle pagine del documento.

Flusso di Elaborazione Dati

Il browser esegue la trasformazione XSLT sul documento XML per generare una pagina HTML. Successivamente, gli script e i fogli di stile richiamati da questa pagina creano l'interfaccia utente interattiva.

Su GIT ho creato una directory **sviluppo** con file utilizzati durante la fase 1.

<https://github.com/davidecaruso03/codificaditestri/tree/main/sviluppo>

Relativamente alla Fase 1, apprendo

<https://www.pisa.live/tei>

si può vedere la versione che usa la generazione dell'HTML direttamente dal browser a partire da xml + xsl

5. Analisi Tecnica dei Componenti

5.1. Documento TEI XML (`testo.xml`)

Il documento è strutturato secondo gli standard TEI P5 con un `<teiHeader>` contenente i metadati e un `<body>` contenente la trascrizione. La fonte è il Volume 6, Fascicolo n. 144 de "La Rassegna Settimanale di Politica, Scienze, Lettere ed Arti" pubblicato a Roma il 3 ottobre 1880 dalla Tipografia Barbera.

Struttura del TEI Header

- **titleStmt**: Titolo principale e sottotitolo, fondatori della rivista (Leopoldo Franchetti e Sydney Sonnino), trascrittore
- **publicationStmt**: Informazioni sull'edizione digitale, disponibilità e link al progetto
- **seriesStmt**: Contesto dell'edizione elettronica coordinata dal Prof. Angelo Mario Del Grossi
- **notesStmt**: Note sulla digitalizzazione e link al sito della rivista
- **sourceDesc**: Struttura bibliografica completa con 6 analytic (articoli), monogr (rivista), imprint (dettagli pubblicazione)

Contenuto Codificato

Il documento contiene **6 sezioni** (div) che rappresentano diversi articoli e rubriche:

1. **Corrispondenza da Castellammare** - Articolo sul varo della nave da guerra "Italia"
2. **Primavera** - Racconto letterario di R. Fucini
3. **Bibliografia** - Due recensioni di opere letterarie e storiche
4. **Notizie dai Periodici** - Rassegna di pubblicazioni inglesi e francesi

Codifica Semantica

Il testo include **13 tipologie di fenomeni notevoli** identificati e codificati:

- **persName**: Nomi di persona (es. Barnaby, Ferrari, Comba, Valdo)
- **placeName**: Luoghi geografici (es. Castellammare, Sassoferato, Verona, Lione)
- **name[@type='ship']**: Nomi di navi (es. Italia, Duilio, Thunderer, Devastation)
- **orgName**: Organizzazioni (es. marina italiana, marina britannica, Marina Germanica)
- **date**: Date e periodi temporali
- **title**: Titoli di opere, riviste, articoli
- **q**: Citazioni e discorsi diretti
- **emph**: Enfasi e frasi evidenziate
- **hi[@rend='exclamation']**: Esclamazioni
- **term**: Termini tecnici
- **roleName**: Ruoli e professioni
- **country**: Nomi di stati
- **ident**: Identificatori di oggetti

Sezione Facsimile

La sezione `<facsimile>` definisce **6 superfici** (surface) corrispondenti a 6 pagine digitalizzate, con **un totale di 466 zone** (righe di testo):

- Ogni riga di testo (`<1>`) nel body è collegata a una zona tramite l'attributo `@facs`
- Ogni zona è definita con coordinate rettangolari: `@u1x`, `@uly` (angolo superiore sinistro), `@lrx`, `@lry` (angolo inferiore destro)

Stato della Mappatura:

- **Pagina 1** ([Pagina9.jpg](#)): 53 zone completamente mappate con coordinate precise
- **Pagina 2** ([Pagina10.jpg](#)): 133 zone mappate con coordinate precise
- **Pagina 3** ([Pagina11.jpg](#)): 113 zone mappate con coordinate precise
- **Pagina 4** ([Pagina17.jpg](#)): 107 zone mappate con coordinate precise
- **Pagina 5** ([Pagina18.jpg](#)): 31 zone mappate con coordinate precise
- **Pagina 6** ([Pagina21.jpg](#)): 29 zone mappate con coordinate precise

Tutte le pagine sono quindi **completamente interattive** con mappatura precisa delle coordinate.

5.2. Foglio di Trasformazione XSLT (`tei_transform.xsl`)

Il foglio XSLT 1.0 è il cuore della trasformazione da TEI XML a HTML interattivo.

Funzione Principale

La trasformazione XSLT ha lo scopo fondamentale di **convertire il documento TEI XML strutturato in una pagina HTML completamente funzionale** che può essere visualizzata direttamente in un browser. Non si limita a una semplice conversione statica, ma genera dinamicamente:

- Struttura HTML con layout a due colonne
- Array JavaScript con coordinate delle zone estratte dall'XML
- Collegamenti dinamici tra immagini e testo tramite ID sincronizzati
- Metadati formattati nella sezione descrittiva

- Elementi semanticci con tooltip informativi

Template Principali

1. Template Root (`match="/"` , righe 66-206)

- Genera l'intero documento HTML5
- Crea struttura `<head>` con riferimenti a CSS
- Costruisce sezione descrittiva con metadati estratti da `teiHeader`
- Genera container principale con layout a due colonne
- Loop dinamico su tutte le `<surface>` del facsimile per creare immagini e image map
- Genera script JavaScript con array `zonesData` popolato dinamicamente

2. Template per Metadati (righe 34-61, 284-309)

- `render-metadata-item-new` : Visualizza metadati semplici come "label: content" (righe 34-42)
- `render-transcriber-link` : Gestisce link ipertestuali per trascrittore e coordinatore (righe 43-61)
- `render-metadata-item-simple` : Template condizionale che mostra metadati solo se presenti (righe 284-295)
- `render-metadata-item-link-simple` : Crea link esterni per progetto digitale (righe 297-309)
- Estrae informazioni da `titleStmt`, `publicationStmt`, `seriesStmt`, `sourceDesc`

3. Template per Struttura Documento (righe 207-282)

- `tei:head` : Trasforma titoli in `<h3>` con classi CSS
- `tei:p` : Wrapper per paragrafi con classe `.tei-paragraph`
- `tei:l` : **Template complesso** per righe di testo con logica condizionale:
 - Distingue righe di capitolo (`seg[@type='chapter']`) da righe normali
 - Gestisce attributi `@rend` (indent, center, right) con mapping a classi CSS
 - Crea ID sincronizzati con zone: `id="{substring-after(@fac, '#')}"`
 - Gestisce interruzioni di riga (`
`) in base al tipo di rendering
- `tei:seg[@type='chapter']` : Trasforma in `` per enfasi

4. Template per Elementi Semanticci (righe 310-420)

- **13 template specializzati** per fenomeni notevoli:
 - `tei:persName` → ``
 - `tei:placeName` → ``
 - `tei:name[@type='ship']` → ``
 - `tei:orgName` → ``
 - `tei:date` → ``
 - `tei:title` → ``
 - `tei:q` → `<q class="quoted-text" title="Citazione">`
 - `tei:emph` → `<em class="emph" title="Enfasi">`
 - `tei:hi[@rend='exclamation']` → ``
 - `tei:term` → ``
 - `tei:roleName` → ``
 - `tei:country` → ``
 - `tei:ident` → ``
- Ogni template aggiunge classe CSS per styling e attributo `title` per tooltip

5. Template per Rendering Visuale (righe 402-420)

- `tei:hi[@rend='bold']` → `` (grassetto)
- `tei:hi[@rend='italic']` → `<i class="rend-italic">` (corsivo)

6. Template Fallback (riga 426)

- `text()` : Cattura tutti i nodi di testo senza template specifico e li rende come contenuto semplice

Generazione Dinamica Zone

Il blocco più critico dell'XSLT è la generazione dell'array JavaScript `zonesData` (righe 184-201):

```
<xsl:for-each select="/tei:TEI/tei:facsimile/tei:surface/tei:zone">
    <xsl:variable name="zoneId" select="substring-after(@xml:id, 'ZONE_')"/>
    <xsl:variable name="ulx" select="@ulx"/>
    <xsl:variable name="uly" select="@uly"/>
    <xsl:variable name="lrx" select="@lrx"/>
    <xsl:variable name="lry" select="@lry"/>
    { id: '<xsl:value-of select="$zoneId"/>',
      coords: [<xsl:value-of select="$ulx"/>, <xsl:value-of select="$uly"/>,
                <xsl:value-of select="$lrx"/>, <xsl:value-of select="$lry"/>] }
</xsl:for-each>
```

Questo codice:

- Itera su tutte le 466 zone di tutte le 6 pagine
- Rimuove il prefisso "ZONE_" dall'ID per creare ID pulito (es. ZN_P1_R1)
- Estrae le 4 coordinate (ulx, uly, lrx, lry)
- Genera oggetto JavaScript con struttura `{ id: '...', coords: [...] }`
- Aggiunge virgola tra elementi tranne l'ultimo (`<xsl:if test="position() != last()">,</xsl:if>`) in maniera da generare un array JavaScript valido

Loop Multi-Pagina Dinamico

Per supportare qualsiasi numero di pagine (righe 152-166):

```
<xsl:for-each select="/tei:TEI/tei:facsimile/tei:surface">
    <div class="page-wrapper">
        
        <map>
            <xsl:for-each select=".//tei:zone">
                <area shape="rect" coords="{@ulx},{@uly},{@lrx},{@lry}"
                      alt="Riga di testo" href="javascript:void(0);"
                      data-zone-id="{@xml:id}"/>
            </xsl:for-each>
        </map>
    </div>
</xsl:for-each>
```

Questo genera automaticamente:

- Un `page-wrapper` per ogni `<surface>`

- Immagine con ID dinamico basato su `position()` (page1-image, page2-image, ecc.)
- Image map con aree cliccabili per ogni zona della pagina
- Attributi data per sincronizzazione JavaScript

Caratteristiche Tecniche

- **XSLT 1.0:** Massima compatibilità con tutti i processori XSLT
- **Output HTML indent:** Codice generato leggibile e ben formattato
- **Namespace TEI:** Gestione corretta con prefisso `tei:`
- **XPath precisi:** Selezione accurata di elementi con predicati
- **Template vuoti:** Template vuoti per `tei:availability` per escludere contenuto
- **Modularità:** Template separati per ogni tipo di elemento facilitano manutenzione

5.3. Script JavaScript (`tei_transform.js`)

Lo script, scritto in **JavaScript ES5** per massima compatibilità, gestisce l'intera logica interattiva del sistema.

Architettura dello Script

1. Utility Functions (righe 29-36)

- Rilevamento dispositivi iOS per gestione specifica di problemi di rendering

2. Inizializzazione e Gestione Eventi (righe 38-78)

- `window.onload` : Inizializzazione al caricamento pagina
- `window.addEventListener('resize')` : Gestione ridimensionamento con timeout di 100ms per ottimizzazione
- Ricalcolo automatico delle coordinate delle zone in risposta al ridimensionamento del browser

3. Creazione Dinamica delle Zone Interattive (righe 80-139)

- `createTestZonesDirectly()` : Funzione principale che itera su tutti i page-wrapper
- Supporto automatico per qualsiasi numero di pagine (scalabile)
- Rimozione overlay esistenti per evitare duplicati
- Creazione di due layer: zone visibili e aree invisibili

4. Evidenziazione del Testo (righe 141-176)

- `highlightText(zoneId)` : Sincronizzazione tra immagine ↔ testo
- Rimozione evidenziazione precedente (single selection)
- Scroll automatico con animazione smooth per centrare riga evidenziata
- Gestione speciale per dispositivi iOS

5. Controlli Interfaccia Utente (righe 178-309)

- `ToggleHighlights()` : Mostra/nasconde colori di 13 tipologie di fenomeni notevoli
- `ToggleZoneOverlay()` : Gestisce doppio sistema di interazione (zone visibili/invisibili)
- Feedback visivo con cambio colore pulsanti (blu=attivo, rosso=disattivo)
- Supporto multi-pagina con applicazione modifiche a tutte le pagine

6. Sistema di Scaling Responsive (righe 311-429)

- `calculateScaleFactor()` : Calcola rapporto tra dimensioni naturali e visualizzate
- `scaleZoneCoordinates()` : Applica fattore di scala a coordinate originali

- `createFallbackZonesForPage()` : Crea zone visibili con overlay rosso
- Gestione effetti hover per feedback visivo
- Etichette zone con ID per debug

7. Gestione Zone Dinamiche Multi-Pagina (righe 431-495)

- `generateDynamicZonesForPage()` : Filtra zone per pagina specifica usando pattern matching
- Conversione formato da array coordinati a oggetti con proprietà nominate
- Supporto automatico per qualsiasi numero di pagine

8. Aree Invisibili per Click (righe 497-558)

- `createInvisibleClickAreasForPage()` : Crea layer trasparente per mantenere interattività quando overlay è nascosto
- Stesse coordinate delle zone visibili ma completamente trasparenti
- Gestori eventi per click e identificazione zone

9. Ottimizzazione Layout (righe 560-627)

- `adjustMainContainerHeight()` : Calcola e imposta altezza ottimale del container
- Gestione precisa di margini e padding
- Workaround specifico per problemi di resize su iOS/iPad

Variabili Globali

- `ZoneEvidenziate` (booleano): Traccia stato zone overlay (visibili/nascoste)
- `LastSelectedLine` : Riferimento all'ultima riga selezionata (per workaround iOS)
- `zonesData` : Array generato dinamicamente da XSLT contenente tutte le zone con coordinate

Caratteristiche Tecniche

- **Compatibilità universale:** JavaScript ES5 funziona su tutti i browser moderni e legacy
- **Performance ottimizzata:** Timeout e debouncing per eventi frequenti (resize)
- **Accessibilità:** Supporto per screen reader e navigazione da tastiera
- **Responsive design:** Adattamento automatico a qualsiasi dimensione schermo
- **Cross-platform:** Gestione specifica per problemi iOS/iPad

5.4. Foglio di Stile CSS (`tei_transform.css`)

Il foglio di stile definisce la presentazione completa del sistema con un design moderno e responsive.

Layout Principale

- **Layout Flexbox a due colonne:** Immagini facsimile a sinistra, testo codificato a destra
- **Responsive design:** Adattamento automatico con `flex: 1` per distribuire spazio equamente
- **Container principale:** `max-width: 1600px`, centrato con `margin: 0 auto`
- **Altezza dinamica:** Calcolata da JavaScript per ottimizzare uso spazio verticale
- **Scroll indipendente:** Ogni colonna con `overflow-y: auto` per navigazione separata

Sistema di Codifica a Colori

Implementa un sistema di 13 colori distintivi per i fenomeni notevoli:

1. `persName` - `#0056b3` (blu) - Nomi di persona

2. **placeName** - #28a745 (verde) - Luoghi geografici
3. **name** - #ff00ff (magenta) - Nomi propri/navi
4. **orgName** - #663399 (viola scuro) - Organizzazioni
5. **date** - #0000ff (blu puro) - Date
6. **title** - #ffa500 (arancione) - Titoli di opere
7. **quoted-text** - #303030 (grigio scuro) + italic - Citazioni
8. **emph** - #e05d00 (arancione scuro) - Enfasi
9. **exclamation** - #dc3545 (rosso) - Esclamazioni
10. **term** - #c82333 (rosso cremisi) - Termini tecnici
11. **roleName** - #ff69b4 (rosa acceso) - Ruoli
12. **country** - #20e040 (verde brillante) - Paesi
13. **ident** - #60a040 (verde oliva) - Identificatori

Ogni elemento ha:

- `font-weight: bold` per maggiore visibilità
- `cursor: help` per indicare tooltip disponibile
- `position: relative` per posizionamento tooltip

Sistema di Tooltip Personalizzati

- **Tooltip CSS puro:** Implementati con pseudo-elementi `::before` e `::after`
- **Posizionamento intelligente:** Sopra l'elemento con freccia direzionale
- **Stile uniforme:** Background scuro (#333), testo bianco, padding 6px 10px
- **Tooltip colorati:** Ogni tipologia ha colore specifico che corrisponde al colore del testo
- **Responsive:** `max-width: 250px`, `word-wrap: break-word` per testi lunghi
- **Effetto smooth:** Appare solo su hover con transizione automatica

Classi di Rendering

- **rend-indent:** Rientro 2em per paragrafi indentati
- **rend-indent-2:** Rientro 4em per indentazione doppia
- **rend-align-right:** Allineamento a destra con `display: block`
- **rend-align-center:** Centratura testo con `display: block`
- **rend-center-indent:** Centratura con spostamento 12em per effetto speciale
- **rend-bold:** Grassetto per elementi `<hi rend="bold">`
- **rend-italic:** Corsivo per elementi `<hi rend="italic">`

Stili di Evidenziazione

- **highlight:** Background #ffe5b4 (beige chiaro), bordo #ff6b35 (arancione), padding 2-4px
- **hover sulle righe:** Background #f0f8ff (azzurro tenue) per feedback visivo
- **transition:** `background-color 0.3s ease` per animazioni fluide

Controlli UI

- **toggle-button:** Pulsanti blu (#007bff) con hover più scuro (#0056b3)
- **controls-container:** Bordo superiore separatore, padding 5px
- **Cambio colore dinamico:** JavaScript modifica background per indicare stato attivo/inattivo

Ottimizzazioni

- **box-sizing: border-box:** Su tutti gli elementi per calcolo dimensioni preciso
- **min-height e height:** Gestione altezza con vincoli minimi per stabilità layout
- **border-radius:** 4-8px per angoli arrotondati moderni
- **box-shadow:** `0 4px 8px rgba(0,0,0,0.1)` per profondità visiva
- **line-height ottimizzata:** 1.8 per paragrafi, 1.2 per righe interattive

Gestione Stati Speciali

- **highlights-off:** Classe che disabilita `pointer-events` quando fenomeni sono nascosti
- **hidden:** `display: none !important` per elementi completamente nascosti
- **quotes personalizzate:** `q { quotes: "" "";` per evitare virgolette automatiche

6. Statistiche del Progetto

Dimensioni del Codice (Fase 1)

Componente	Righe di Codice	Dimensione	Linguaggio
<code>testo.xml</code>	1.078	~91 KB	TEI XML
<code>tei_transform.xsl</code>	432	~17 KB	XSLT 1.0
<code>tei_transform.js</code>	628	~24 KB	JavaScript ES5
<code>tei_transform.css</code>	605	~9 KB	CSS3
<code>tei_all.dtd</code>	6.171	~359 KB	DTD
<code>index.html</code>	605	~1 KB	HTML5
TOTALE	9.519	~498 KB	-

Contenuto Codificato

- **Pagine digitalizzate:** 6 facsimili JPG
- **Totale zone interattive:** 466 righe di testo mappate
- **Fenomeni notevoli codificati:** 13 tipologie diverse
- **Sezioni documento:** 6 div (articoli e rubriche)
- **Metadati:** 6 analytic + 1 monogr nella biblStruct

Distribuzione Zone per Pagina

Pagina	Immagine	N. Zone	Stato
1	Pagina9.jpg	53	Completa
2	Pagina10.jpg	133	Completa
3	Pagina11.jpg	113	Completa
4	Pagina17.jpg	107	Completa
5	Pagina18.jpg	31	Completa
6	Pagina21.jpg	29	Completa

Tecnologie e Standard

- **Standard TEI:** P5 versione 4.10.2
- **Validazione:** Conforme a DTD `tei_all.dtd`
- **XSLT:** Versione 1.0 per compatibilità universale
- **JavaScript:** ES5 per supporto browser legacy
- **CSS:** CSS3 con Flexbox
- **Browser compatibili:** Tutti i browser moderni (Chrome, Firefox, Safari, Edge) + browser legacy con JavaScript ES5

7. Funzionalità Implementate

Interattività

- **Zone cliccabili dinamiche** - Overlay rosso con coordinate scalate responsive
- **Sincronizzazione tra immagine e testo** - Click su zona → evidenzia testo
- **Scroll automatico** - Centra automaticamente la riga selezionata
- **Doppio sistema di interazione** - Zone visibili/invisibili con toggle
- **Effetti hover** - Feedback visivo su zone e righe di testo

Visualizzazione

- **Layout responsive a due colonne** - Immagini e testo affiancati
- **Scroll indipendente** - Navigazione separata per colonne
- **Multi-pagina dinamico** - Supporto automatico per N pagine
- **Adattamento finestra** - Ricalcolo coordinate al resize
- **Tooltip informativi** - 13 tipi di fenomeni con descrizioni

Codifica Semantica

- **13 tipologie di fenomeni notevoli** con colori distintivi
- **Toggle fenomeni** - Mostra/nasconde evidenziazioni colorate
- **Metadati strutturati** - TEI Header completo con biblStruct
- **Attributi di rendering** - Gestione indent, center, right, italic, bold
- **Citazioni e enfasi** - Markup semantico per elementi letterari

Accessibilità

- ARIA labels - Supporto per screen reader
 - Tooltip descrittivi - Spiegazione di ogni fenomeno notevole
 - Cursor: help - Indicatore visivo per elementi interattivi
 - Contrast ratio - Colori scelti per leggibilità
-

8. Verifica e generazione progetto definitivo con Saxon (Fase 2)

Tra i requisiti del progetto c'è la generazione tramite **Saxon** dell'HTML.

Nella fase di creazione del progetto (Fase 1) ho utilizzato la generazione dell'HTML direttamente dal browser a partire da xsl.

Questo ha consentito di velocizzare la fase di sviluppo, consentendo di modificare e testare il funzionamento interattivamente e in tempo reale.

Una volta ottenuto il risultato desiderato dal punto di vista della grafica, interattività e rispetto delle caratteristiche desiderate sono passato alla generazione tramite Saxon dell'HTML (Fase 2).

Inizialmente, per una verifica veloce, ho usato il sito: <http://xslttransform.net> verificando che tutto andasse bene sia con diverse versioni di Saxon che con Xalan

Quindi ho usato su PC la versione locale java **saxon-he-12.9**

con il comando

java -jar F:\SaxonHE\saxon-he-12.9.jar -s:testo.xml -xsl:tei_transform.xsl -o:index.html

per verificare la correttezza del file **tei_transform.xsl** e per generare il file **index.html** dell'esempio finale.

Attenzione: per la generazione con Saxon ho creato una copia del file **testo.xml** nella directory Saxon, dalla quale ho rimosso la riga 3 contenente l'inclusione di **tei_transform.xsl** poiché non necessaria per Saxon (che riceve il documento xsl come argomento da riga di comando)

Non ho ricevuto errori dalla trasformazione e ho dunque utilizzato il file generato **index.html**

Ho quindi ritestato il funzionamento e il rispetto dei requisiti W3C con la versione finale che utilizza l'HTML generato da Saxon

Su GIT ho creato una directory **saxon** con i file utilizzati per la trasformazione.

<https://github.com/davidecaruso03/codificaditesti/tree/main/saxon>

e ho creato una di directory **sito** con la versione definitiva del progetto (quella creata con saxon)

<https://github.com/davidecaruso03/codificaditesti/tree/main/sito>

Aprendo

<https://www.pisa.live/tei3>

si può vedere la versione finale funzionante che usa l'HTML generato tramite Saxon

Componenti Principali Fase 2

- **index.html** : Documento HTML ottenuto dalla trasformazione con Saxon.
- **Immagini JPG** : 6 facsimili digitalizzati delle pagine del documento.

Dimensioni del Codice (Fase 2)

Componente	Righe di Codice	Dimensione	Linguaggio
index.html	2.706	~203 KB	HTML5
TOTALE	2.706	~203 KB	-

9. Struttura delle directory

Il progetto è strutturato in quattro directory principali che riflettono le fasi di sviluppo e distribuzione:

```
Progetto/
├── ReadME.md                                # Documentazione principale
│
├── sviluppo/
│   ├── index.html                            # FASE 1: Ambiente di lavoro e sorgenti
│   ├── testo.xml                             # Viewer iframe per sviluppo interattivo
│   ├── tei_all.dtd                           # Sorgente TEI XML principale
│   ├── tei_transform.xsl                      # Validazione schema
│   ├── tei_transform.js                      # Documento XSLT
│   ├── tei_transform.css                      # Logica JavaScript inclusa in .xsl
│   ├── zone.xlsm                            # Foglio di stile CSS incluso in .xsl
│   ├── ReadME.md                            # Excel per mappatura coordinate
│   ├── ReadME.pdf                           # Copia della Documentazione principale
│   ├── Elementi TEI usati.md                # Copia Pdf della Documentazione principale
│   ├── favicon.ico                         # Elenco elementi TEI usati in testo.xml
│   ├── [Immagini JPG]                      # Icona del sito
│   └── [Immagini JPG]                      # Facsimili per lo sviluppo
│
├── saxon/
│   ├── EseguiSaxon.bat                     # FASE 2: Strumenti di generazione
│   ├── testo.xml                           # Script per automatizzare la trasformazione
│   ├── tei_all.dtd                           # Sorgente TEI XML principale
│   ├── tei_transform.xsl                    # Validazione schema
│   ├── index.html                          # Documento XSLT (contenente all'interno css e js)
│   └── VerificaSaxon.txt                  # Output generato da Saxon
│                                         # Descrizione della transcodifica
│
├── xerces/
│   ├── EseguiXerces.bat                   # VALIDAZIONE con Xerces
│   ├── testo.xml                           # Script per invocare la validazione
│   └── tei_all.dtd                           # Sorgente TEI XML principale (senza inclusione di xsl)
│                                         # Validazione schema
│
└── sito/
    ├── index.html                         # VERSIONE FINALE: Deploy
    ├── ReadME.md                           # Versione finale statica (output di Saxon)
    ├── ReadME.pdf                          # Copia della Documentazione principale
    ├── favicon.ico                        # Copia Pdf della Documentazione principale
    └── [Immagini JPG]                      # Icona del sito
    └── [Immagini JPG]                      # Facsimili per il sito web
```

10. Punti di Forza del Progetto

1. Architettura Modulare e Riusabile

- I file di trasformazione, stile e script sono completamente indipendenti dal contenuto specifico
- Possono essere applicati a qualsiasi documento TEI con struttura simile
- Supporto automatico per qualsiasi numero di pagine senza modifiche al codice

2. Scalabilità e Performance

- Sistema responsive che si adatta a qualsiasi dimensione schermo
- Ottimizzazioni per ridimensionamento (timeout, debouncing)
- Gestione efficiente di 466 zone interattive

3. Standard e Compatibilità

- Piena conformità allo standard TEI P5 versione 4.10.2
- Validazione DTD completa
- Compatibilità con tutti i browser moderni e legacy

4. User Experience

- Interfaccia intuitiva con controlli chiari
- Feedback visivo immediato su tutte le interazioni
- Doppio sistema di interazione per preferenze utente
- Tooltip informativi per didattica

5. Documentazione

- Codice commentato
 - ReadME esplicativo per utilizzo
-

11. Conclusioni

Questo progetto punta a dimostrare come gli standard TEI P5 possano essere efficacemente combinati con tecnologie web moderne (XSLT, JavaScript, CSS) per creare **strumenti di visualizzazione interattiva** di testi storici e letterari.

Obiettivi Raggiunti

- **Sistema completamente funzionale** con tutte le 6 pagine mappate e interattive
- **Codifica semantica completa** con 13 tipologie di fenomeni notevoli
- **Architettura riusabile** indipendente dal contenuto specifico
- **Performance ottimali** con gestione responsive e adattamento dinamico
- **Documentazione esaustiva** sia nel codice che in questo documento

Competenze Sviluppate

- **Codifica TEI P5:** Strutturazione metadati, markup semantico, facsimile digitali
- **XSLT:** Trasformazioni complesse da XML a HTML con generazione dinamica
- **JavaScript:** Manipolazione DOM, gestione eventi, calcoli geometrici per scaling

- **CSS:** Layout Flexbox, tooltip personalizzati, design responsive
- **Workflow digitale:** Dall'estrazione immagini alla pubblicazione web

Valore Didattico e Scientifico

Il sistema sviluppato rappresenta un esempio di come le Digital Humanities possano valorizzare il patrimonio culturale attraverso tecnologie accessibili. La visualizzazione sincronizzata di facsimile e trascrizione codificata facilita:

- **Studio filologico:** Confronto immediato tra immagine originale e testo
- **Ricerca semantica:** Identificazione rapida di entità nominate
- **Didattica:** Insegnamento dell'analisi testuale e della codifica XML-TEI
- **Disseminazione:** Pubblicazione online di edizioni digitali interattive