



Università degli Studi di Salerno
Facoltà di Scienze Matematiche Fisiche e Naturali

Tesi di Laurea di I livello in
Informatica

Visualizzazione di Open Data come Timeline

Relatore

Prof. Vittorio Scarano

Candidato

Vincenzo Santoro

Anno Accademico 2015-2016

Dediche e ringraziamenti

Ringrazio Andrea e tutto il personale di ISIS Lab per il supporto tecnico durante il lavoro;
la mia famiglia, per il supporto economico e morale durante i miei anni di studio;
il Prof. Vittorio Scarano, per la possibilità datami con il tirocinio in ISIS Lab.
A Miriam, che da quando è entrata nella mia vita mi è sempre restata accanto.



Questa tesi è stata sviluppata in **ISISLab**

Indice

1	Introduzione	1
2	Il caso di studio	3
2.1	Open Data	3
2.2	SPOD	4
2.3	OXWALL	5
2.4	DatalEt-Ecosystem Provider (DEEP)	6
3	Timeline	9
3.1	Cos'è una Timeline	9
3.2	Requisiti	10
3.3	Stato dell'Arte	11
3.3.1	Chronozoom	11
3.3.2	Crossfilter	14
3.3.3	Timeline di Simile Project	15
3.3.4	Google Charts	17
3.3.5	Timemapper	19
3.4	Tecnologie utilizzate	21
3.4.1	Javascript	21
3.4.2	Polymer e Web Components	21
3.4.3	JSON	23
3.5	Timeline JS3	24
3.6	Confronto tra tecnologie	25
3.7	Codice in dettaglio	26
3.7.1	Manipolazione di date	26
3.7.2	Manipolazione di numeri romani	29
3.7.3	La conversione dei dati in formato JSON	32
4	Conclusioni	34

<i>INDICE</i>	iii
Bibliografia	36
A Appendice	38
A.1 Immagini	38
A.2 Listati	44
A.2.1 timeline-datalet	44
A.2.2 timeline-demo	54

Capitolo 1

Introduzione

In questo lavoro di tesi viene realizzato un sistema di visualizzazione per dati aperti (open data) basato sul tempo. Nello specifico viene realizzata una "linea del tempo" (timeline) interattiva in cui l'utente può navigare e da cui egli può ottenere maggiori informazioni sull'evento del quale si sta informando. Il lavoro troverà applicazione nel campo dei beni culturali e nell'archeologia, ma potrà essere usato anche in altri campi, purché i dati utilizzati facciano uso di date (avvenimenti storici, date di nascita e morte di personalità importanti, date di delibere comunali, ...).

La tesi è stata svolta all'interno del progetto europeo "ROUTE-TO-PA" [1] (Raising Open and User-friendly Transparency-Enabling Technologies fOr Public Administrations), il quale mira al miglioramento del coinvolgimento dei cittadini rendendo loro possibile tessere relazioni sociali sugli Open Data, fondando o unendosi a comunità online che condividono interessi comuni per discutere di problemi comuni.

La Timeline è progettata per l'utilizzo nella piattaforma sociale annessa al progetto "SPOD" [2] che permette agli utenti di interagire con le funzionalità classiche di un social network offrendo inoltre la possibilità di visualizzare i dati (datasets) con rappresentazioni grafiche interattive (datalet).

Il progetto, oltre alla rappresentazione dei dati in formato di Timeline, consente di inserire mappe o contenuti multimediali, ed è integrato completamente con l'architettura sottostante e con le tecnologie già disponibili sulla piattaforma SPOD.

L'utente che andrà a creare le timeline non avrà bisogno di alcuna competenza specifica se non quelle normalmente richieste per l'utilizzo della piattaforma SPOD.

Nei capitoli che seguono, verrà illustrato il "caso di studio", con una descri-

zione delle tecnologie con cui si è lavorato, lo "stato dell'arte", che descriverà le tecnologie attualmente disponibili sul mercato per risolvere questa tipologia di problema e in seguito verrà analizzata la soluzione proposta, con particolare attenzione al codice della stessa. In conclusione, alcuni possibili sviluppi futuri del progetto.

Capitolo 2

Il caso di studio

2.1 Open Data

Con "Open Data" (dati aperti) [3] si intendono dati liberamente accessibili da tutti le cui eventuali restrizioni sono l'obbligo di citare la fonte o di mantenere la banca dati sempre aperta. Tale concetto rientra nella filosofia dell'"open government" (governo aperto) secondo la quale la Pubblica Amministrazione dovrebbe essere aperta ai cittadini in termini di trasparenza. Questa corrente di pensiero trova applicazione grazie alle nuove tecnologie dell'informazione e della comunicazione, in particolar modo Internet e i Social network, e affonda le sue radici etiche nella filosofia dell'open source.

Secondo il movimento a favore dell'utilizzo degli Open Data, i dati andrebbero trattati come beni comuni; a sostegno di questa tesi vengono poste diverse argomentazioni, tra cui la convinzione che i dati prodotti dalla pubblica amministrazione, in quanto finanziati da denaro pubblico, debbano ritornare ai contribuenti, e alla comunità in generale, sotto forma di dati aperti e universalmente disponibili cosicchè il cittadino possa fruirne liberamente. Inoltre, essendo i dati necessari per agevolare l'esecuzione di comuni attività umane, secondo i sostenitori degli open data, porre restrizioni sui dati e sul loro riutilizzo limita lo sviluppo della comunità.

Il primo esempio di Open Data in Italia è stato OpenStreetMap, un progetto che dal 2007 al 2010 ha raccolto e pubblicato con licenza aperta gli stradari delle pubbliche amministrazioni locali al fine di creare uno stradario liberamente utilizzabile. Il 18 ottobre 2011 ha aperto <http://www.dati.gov.it/>, un portale italiano dell'Open data sul modello dei datagov anglosassoni. Inizialmente Open Knowledge Foundation Italia e il Centro NEXA su Internet e Società del Politecnico di Torino hanno offerto il repository Ckan, dove

chiunque poteva segnalare i dataset italiani disponibili online, in seguito assorbito dal portale internazionale <https://datahub.io/>.

2.2 SPOD

SPOD (Social Platform for Open Data) consente interazioni sociali tra i cittadini attraverso open datasets provenienti da fonti differenti (dette "dataset providers").

SPOD è una piattaforma sociale e, in quanto tale, offre le tradizionali caratteristiche di un social network ai propri utenti. Gli utenti possono scrivere un messaggio di stato, iniziare discussioni, commentare e mettere un "Mi Piace" ai commenti ed ai messaggi di stato degli altri utenti. Come in ogni social network, si possono inviare richieste di amicizia ad altri utenti, scambiare con loro messaggi privati e vedere cosa stanno facendo.

In aggiunta a queste caratteristiche di base, SPOD offre nuove funzionalità come allegare visualizzazioni di datasets (chiamate datalets) ai commenti, "Stanze pubbliche" per la discussione (che assistono l'utente durante la navigazione all'interno dell'albero dei commenti), uno "Spazio privato" dove è possibile fare pratica e costruire datalets privati, e gruppi di collaborazione dove è possibile co-creare dati.

La creazione di una datalet è effettuata da un plug-in sviluppato da ISIS Lab chiamato "DEEP", che assiste e guida l'utente dalla selezione del dataset da cui partire fino alla realizzazione vera e propria della visualizzazione.

2.3 OXWALL

L’architettura di base della piattaforma SPOD è realizzata tramite OXWALL [4], un software gratuito ed open-source estremamente flessibile scritto in PHP/MySQL.

Inizialmente sviluppato da Skalfa LLC, un’organizzazione commerciale, nel 2009, nell’Agosto 2010 è stato rilasciato come software open source. Nel febbraio 2011 Skalfa LLC ha donato la proprietà intellettuale ed i diritti sul marchio registrato Oxwall all’organizzazione no-profit ”Oxwall Foundation” rendendo il progetto esclusivamente Open Source.

L’ultima versione stabile è la 1.8.4, rilasciata il 25 Luglio 2016, e richiede un web server in grado di eseguire PHP 5.3 o superiore, Apache 2 o superiore e MySQL 5.0 o superiore. Dalla versione 1.6 del Gennaio 2014, Oxwall ha una sua versione per navigazione da dispositivi mobili, ottimizzata per la visualizzazione sulla maggior parte dei browser per cellulari attualmente disponibili sul mercato.

Il nucleo di base di Oxwall contiene le caratteristiche base di un qualsiasi social network, come il caricamento e la condivisione di contenuti, richieste di amicizia, pagine di profilo per i singoli utenti, personalizzazione del layout delle pagine e amministrazione degli utenti e dei contenuti. Oxwall fornisce diversi temi di base che sono personalizzabili dall’amministratore del sito.

Le funzionalità di base di Oxwall possono essere estese tramite plug-in nativi o di terze parti forniti gratuitamente o a pagamento nel negozio dedicato: questi forniscono alcune funzionalità come i messaggi privati tra gli utenti (chat), newsfeed stile Facebook con commenti e ”Mi Piace” e la possibilità di caricare foto con parole chiave, voti e commenti. I Plug-In sono implementati tramite l’architettura Model-View-Controller (MVC) con una pagina HTML che definisce la parte grafica (View) del plug-in tramite il motore di template Smarty e la logica di business delle applicazioni (il Controller) scritto in PHP.

Diverse funzionalità di SPOD descritte in precedenza sono implementate tramite plug-in sviluppati appositamente per la piattaforma, come l’Agorà, la Co-creazione, le stanze private e la gestione degli Open Data.

2.4 DataEt-Ecosystem Provider (DEEP)

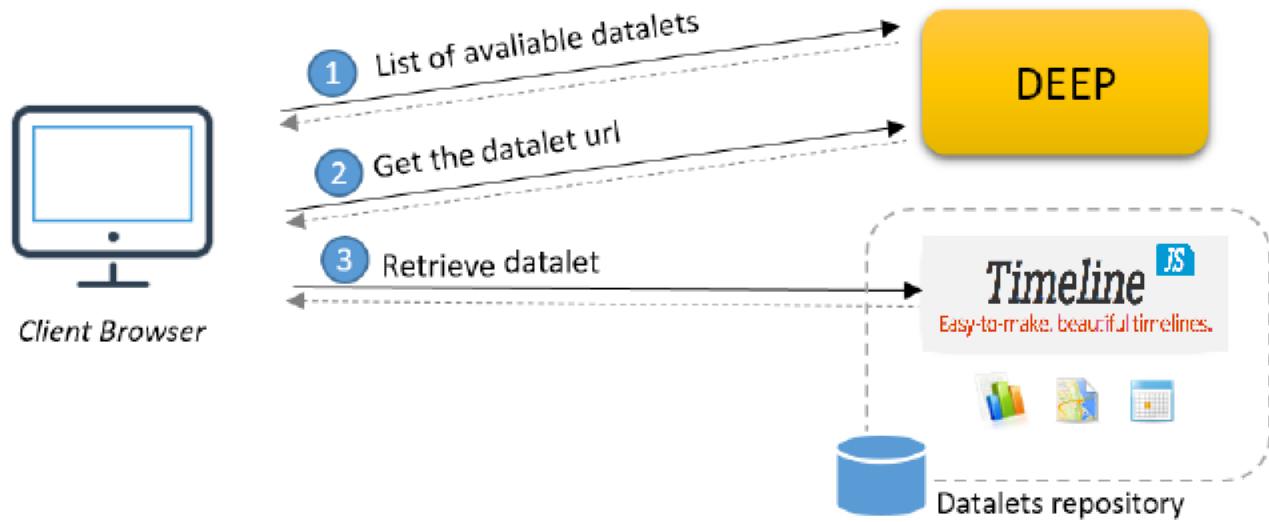


Figura 2.1: Architettura di DEEP

DEEP [5] è stato sviluppato come un'applicazione Restful (REpresentational State Transfer), esso fornisce una lista dei datalet disponibili e il mapping attraverso i nomi delle visualizzazioni e la loro URL all'interno del deposito delle Web Components. Esso è alla base dell'architettura di. Sia DEEP che il deposito delle Web Components sono stati progettati per essere estensibili: possono memorizzare tutte le richieste di visualizzazione e, in futuro, potranno fornire statistiche aggregate sia sulle preferenze degli utenti che sui dati o sulle visualizzazioni. Ad esempio, potrebbero elencare le visualizzazioni di datalet più popolari, i dataset più utilizzati, le visualizzazioni più popolari per un dataset in particolare, il campo più utilizzato per un dataset in particolare e così via.

Il ciclo di lavoro di DEEP per includere una Web Component in una pagina Web, mostrato in figura 2.2, comprende tre servizi principali:

- La pagina Client che richiede di accedere ai servizi offerti da DEEP;
- DEEP;
- La Web Component.

All'inizio la pagina del Client invia una richiesta a DEEP per una specifica Web Component.

A questo punto DEEP risponde con le informazioni necessarie per iniettare la datalet nella pagina.

Come ultima azione, il Client recupera la Web Component dal deposito di DEEP e la inserisce nella pagina.

Le figure A.3, A.4 e A.5 descrivono dal punto di vista dell'utente la creazione di una datalet tramite la procedura guidata di DEEP.

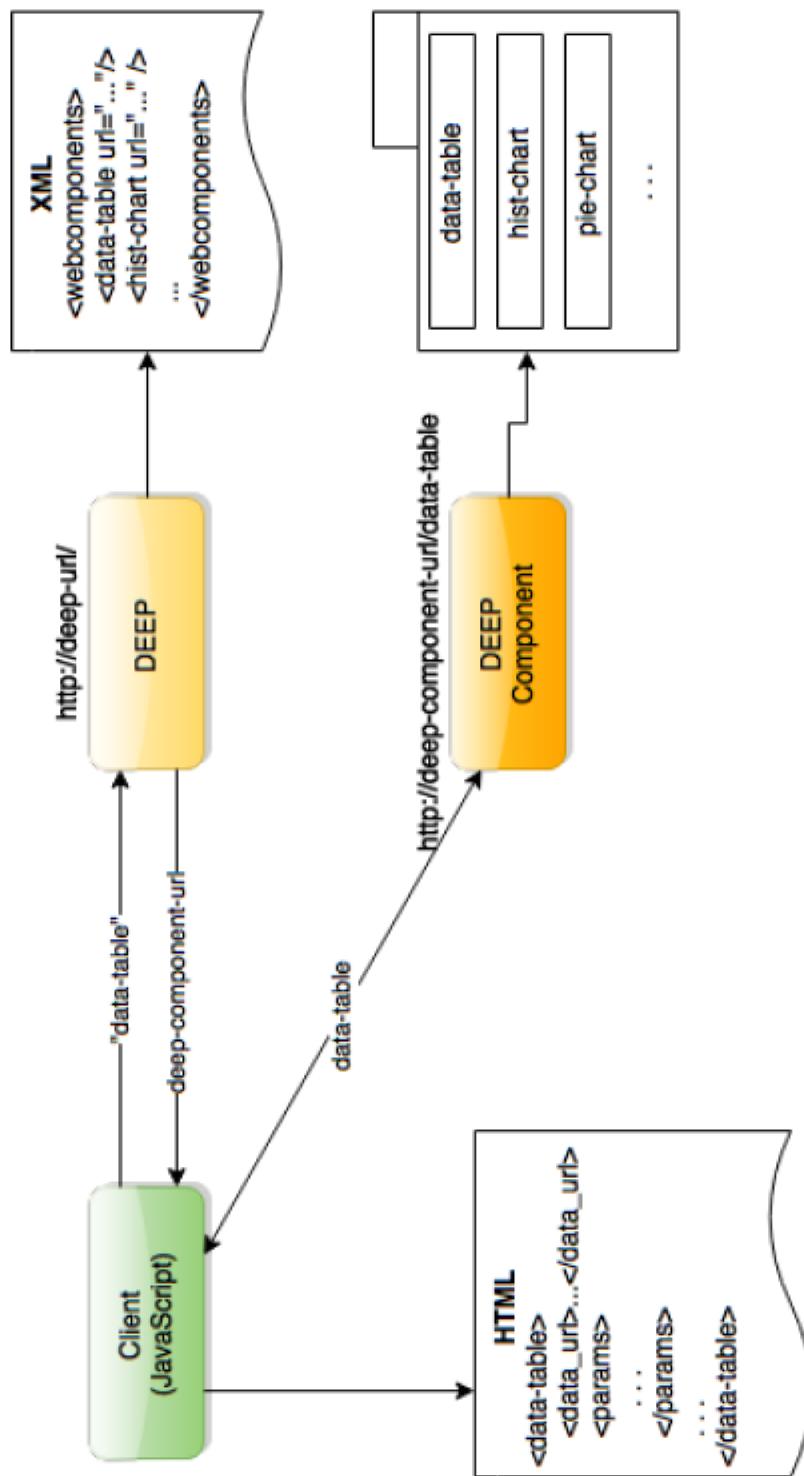


Figura 2.2: Work Cycle (ciclo di lavoro) di DEEP

Capitolo 3

Timeline

3.1 Cos'è una Timeline

Una Timeline (letteralmente "linea del tempo") è uno modo di rappresentare visivamente eventi in ordine cronologico, con l'ausilio di etichette o colori. Oltre a rappresentare eventi localizzati in una singola unità di tempo (che può essere un giorno, un mese o un anno), è possibile rappresentare anche eventi prolungati (su più giorni o su più anni) con la creazione di "intervalli". Inizialmente la grandezza di una Timeline era limitata dallo spazio disponibile su un singolo foglio di carta o in alcuni casi dallo spazio offerto dalla tela; non di rado infatti le Timeline erano realizzate da artisti o comunque da soggetti con abilità pittoriche.

In riferimento all'appendice 1, si pensi al caso di "A New Chart of History", disegnata dal filoso, chimico e teologo Joseph Priestley nel 1765.^{A.1}

Con l'avvento delle nuove tecnologie, le Timeline non sono più relegate ad un singolo foglio di carta, consentendo di rappresentare molti più eventi con maggiore precisione e con intervalli temporali sempre maggiori. La Big History, ad esempio, utilizza intensivamente le Timeline per raccogliere e divulgare tutto ciò che è accaduto dal Big Bang fino ai giorni nostri in ogni campo possibile della conoscenza umana. Per Timeline di così grandi dimensioni, è ovviamente indispensabile un alto livello di interattività con l'utente, che deve essere in grado di navigare tra le date con semplicità: devono essere possibili salti tra date molto distanti tra loro e deve essere possibile zoomare su una specifica porzione della Timeline su cui l'utente ha interesse.

3.2 Requisiti

Di seguito si riportano i requisiti che sono stati presi in esame nella scelta della tecnologia da utilizzare.

Requisiti funzionali:

- Supporto per le immagini: la Datalet deve mostrare, se disponibile, una foto del luogo, evento o bene culturale a cui si riferisce.
- Possibilità di definire un intervallo per date incerte: nel caso in cui non sia possibile associare ad un luogo o ad un bene culturale una datazione precisa, deve essere possibile definire un intervallo della data approssimativa della sua datazione e mostrarlo all'utente.
- Possibilità di selezionare un dato intervallo di tempo dalla Timeline: dato un dataset, deve essere possibile consultare solo una porzione del Dataset selezionando un intervallo di tempo più ristretto rispetto a quello di default della Datalet o in alternativa deve essere possibile effettuare uno zoom su quella porzione di timeline.
- Possibilità di cliccare sugli eventi per ottenere maggiori informazioni (balloon): cliccando su uno degli eventi rappresentati dalla Datalet devono comparire maggiori informazioni legate all'evento, luogo o bene culturale su cui si è cliccato.
- Geo localizzazione: la Datalet dovrebbe permettere di associare alle coordinate temporali anche coordinate geografiche visualizzabili su una mappa associata.

Requisiti di stabilità:

- Open source: la tecnologia scelta dovrà essere gratuita e il codice dovrà essere disponibile liberamente per le eventuali modifiche.
- Ultima release stabile: l'ultima release stabile dovrebbe essere recente, o per lo meno, il software dovrebbe avere frequenti commit su GitHub a testimonianza di un software "vivo" e con un supporto continuo nel tempo.

Requisiti non funzionali:

- Facilità d'uso per l'utente, che dovrà affrontare la creazione della Timeline con la stessa semplicità con cui vengono creati le altre datalet su SPOD.
- Facilità di consultazione per l'utente, che dovrà poter interpretare facilmente i dati contenuti nella Datalet e capire immediatamente come ottenere maggiori informazioni dalla Datalet (nello specifico cliccando sull'evento a cui è interessato).

3.3 Stato dell'Arte

In questa sezione verranno presentate le tecnologie attualmente disponibili sul mercato che sono state prese in esame nello sviluppo del Plug-in e ne verranno elencati i punti di forza, che sono stati fondamentali nella scelta dei requisiti che hanno determinato la tecnologia da cui partire, e i punti deboli che ne hanno comportato l'esclusione dal progetto. Per tutte le tecnologie di seguito elencate, il codice sorgente è liberamente accessibile su GitHub.

3.3.1 Chronozoom

Chronozoom [6] è un progetto open source della Outercurve Foundation in grado di fornire efficaci rappresentazioni interattive di successioni di eventi su una scala temporale che va dal Big Bang al tempo presente (Big History). Il suo approccio alla visualizzazione della storia è stato paragonato all'approccio con cui Google Maps visualizza i dati geografici. Ideato dal geologo Walter Alvarez insieme allo studente Roland Saekow nel 2010, il progetto è stato finanziato e supportato da Microsoft Research Connections in collaborazione con l'Università della California di Berkeley, l'università di Mosca, l'Information School dell'Università di Washington (iSchool), e il Center for Web and Data Science (WDS).

Si tratta di un software particolarmente apprezzato in ambito educativo per l'apprendimento interattivo della storia e per la possibilità di creare le proprie Timeline, motivo per cui nel 2013 ha ricevuto il titolo di «Best Educational Software» al South by Southwest (SXSW).

Tra le caratteristiche offerte troviamo il supporto per immagini e testi, l'inserimento di video da piattaforme di condivisione video (come YouTube) e la possibilità di restringere la Timeline solo a determinate tipologie di eventi o periodi; ogni sezione della Timeline può essere a sua volta suddivisa

in altre sotto-sezioni liberamente, non c'è un limite alla profondità di tali sotto-sezioni. Consente di definire intervalli di tempo ed offre un minimo supporto per le coordinate geografiche, che vengono utilizzate per realizzare sotto-sezioni tematiche. Offre anche la possibilità di copiare parte della Timeline o di condividerla su Twitter.

Il sito su cui è in esecuzione ChronoZoom è open source e sfrutta uno stack di tecnologie Microsoft insieme ad HTML5, mentre le Timeline interattive sfruttano le canvas di HTML5, LESS e CSS3, più JavaScript orientato agli oggetti e jQuery. Le grandi quantità di dati vengono processate dal Software Seadragon per una visualizzazione veloce indipendentemente dalla mole dei dati. Dal punto di vista del client, il risultato è una Single Page Application (SPA).

Il lato server è scritto in C# ed è eseguito su .NET 4.0 con un framework ASP.NET MVC, una REpresentational State Transfer JSON API con Windows Communication Foundation (WCF), un data entity layer, e un Server Microsoft SQL database back-end. Il sito è progettato per essere eseguito su un Microsoft web server (IIS o Azure) e utilizza un Azure Access Control Service per gestire i log-in tramite account Microsoft, Google o Yahoo.

Una versione 1.0 del progetto venne rilasciata nel 2010 durante una serie di lezioni tenute da Álvarez all'Università di Berkeley, California. In seguito una versione beta 2.0 è stata rilasciata nel 2012.

Non è in sviluppo attivo dalla seconda metà del 2015, motivo per cui, nonostante l'elevata interattività del sistema, si è deciso di escluderlo.

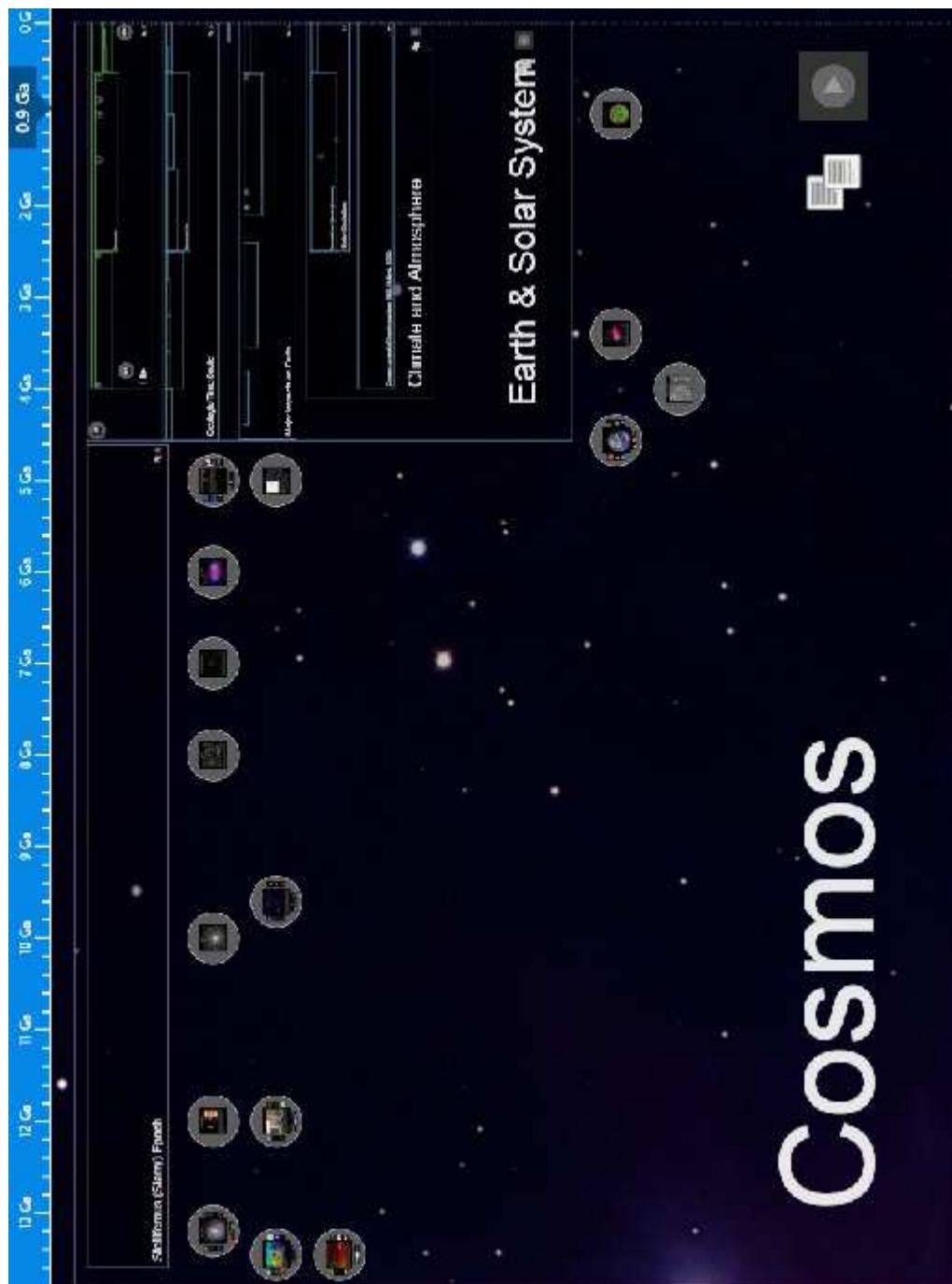


Figura 3.1: Screenshot di una Timeline creata con Chronozoom

3.3.2 Crossfilter

Crossfilter [7] è una libreria JavaScript che permette di esplorare e selezionare nel browser grandi quantità di record da un dataset. Supporta ricerche su dataset contenenti oltre un milione di record e consente di manipolare i dataset in maniera estremamente fluida, ad esempio restringendo gli eventi a determinati giorni della settimana (weekend o giorni di lavoro), determinati mesi o anche di filtrare gli eventi in base agli orari. I dati non sono rappresentati sulla Timeline ma in una tabella separata, la Timeline agisce esclusivamente come filtro per i dati nella tabella.

Inizialmente realizzata per l'azienda statunitense di servizi finanziari Square, Inc., è ora gestita da una Crossfilter Organization composta da volontari. Square, Inc. considera la versione 1.3.14 (Giugno 2016) come l'ultima stabile ed ha dichiarato il software come non più in sviluppo attivo anche se in seguito Crossfilter è stato integrato nella libreria per applicazioni finanziarie dc.js, quest'ultima costantemente aggiornata.

Non supportando immagini o coordinate geografiche, questa tecnologia è stata subito scartata.

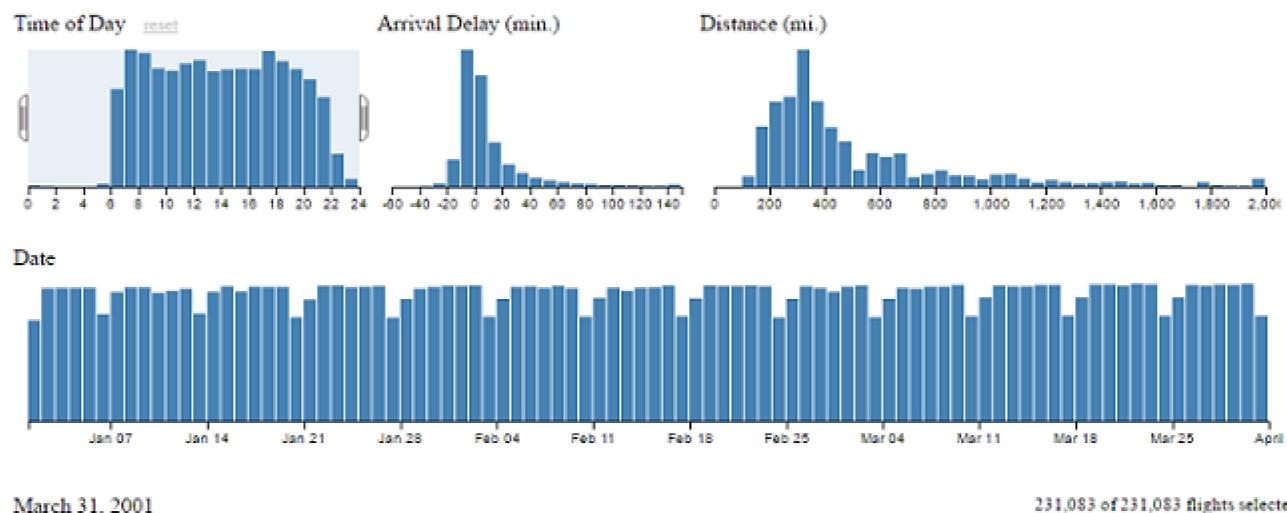


Figura 3.2: Screenshot di una Timeline creata con Crossfilter

3.3.3 Timeline di Simile Project

Parte del SIMILE Project del Massachusetts Institute of Technology [8], questo tool offre la possibilità di realizzare Timeline interattive di eventi. La wiki collegata ad esso lo definisce il "Google Maps" del tempo.

Cliccare su un evento fa comparire un balloon contenente immagini e informazioni addizionali. Una funzionalità interessante è l'inclusione nei Balloon di un tasto «discuti», che inizialmente era stata presa in considerazione per condurre l'utente alla stanza pubblica dell'Agorà in cui si contribuiva allo sviluppo del dataset. Offre supporto alla localizzazione linguistica e tra le lingue supportate si segnala anche l'olandese.

Anche se non consente lo zoom su una porzione della Datalet, Timeline permette di evidenziare i record che contengono una parola nel titolo o nel balloon allegato, inoltre è possibile filtrare i record contenenti una parola data in input eliminando temporaneamente tutti gli altri.

Per rendere la navigazione più agevole in Timeline di grosse dimensioni, è possibile creare dei tasti che evidenzino automaticamente determinate porzioni della Datalet.

Timeline consiste esclusivamente di file statici (librerie Javascript, immagini e file CSS) e per l'installazione sono richiesti Java Runtime e Apache Ant. Non esiste la possibilità di utilizzare coordinate geografiche per la generazione di una mappa associata, questa mancanza ha portato alla decisione di non utilizzarlo.

CAPITOLO 3. TIMELINE

16

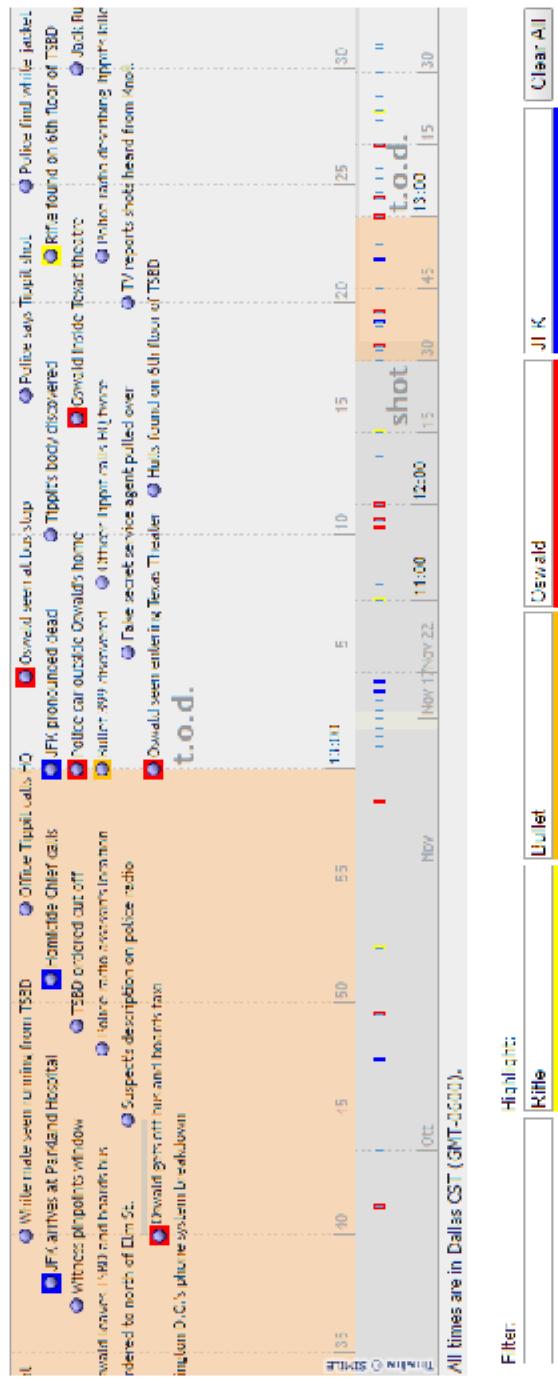


Figura 3.3: Screenshot di una Timeline creata con Timeline di Simile Project

3.3.4 Google Charts

Tra i grafici forniti da Google figurano anche le Timeline [9], per le quali sono fornite diverse opzioni di personalizzazione, come etichette e colori. Supporta l'uso di intervalli di tempo e cliccando su un evento compaiono dei balloon contenenti informazioni addizionali. Tali etichette non possono contenere immagini.

Per accedere alle API di Google Charts basta del semplice codice Javascript per caricare la libreria e generare i dati del grafico, che viene disegnato tramite HTML5 e Scalable Vector Graphics (SVG) per garantire la compatibilità cross-browser e cross-platform (iOS e Android). Non sono richiesti plug-in aggiuntivi. I dati sono memorizzati in una classe DataTable su cui possono essere effettuate Query tramite una classe fornita dalle API.

Segue ora un breve esempio per mostrare come si può creare un semplice grafico a torta tramite Google Charts.

```

1  <!DOCTYPE html>
<html>
<head>
    <script type="text/javascript" src="https://www.
        google.com/jsapi"></script>
    <script type="text/javascript">
        google.load('visualization', '1.0', {'packages
            ':['corechart']});
        google.setOnLoadCallback(drawChart);
        function drawChart() {
            var data = new google.visualization.
                DataTable();
            data.addColumn('string', 'Topping');
            data.addColumn('number', 'Slices');
            data.addRows([
                ['Mushrooms', 3],
                ['Onions', 1],
                ['Olives', 1],
11             ['Zucchini', 1],
                ['Pepperoni', 2]
            ]);
            var options = {'title': 'How Much Pizza I
                Ate Last Night',
                'width':400,
                'height':300};
21

```

```

        var chart = new google.visualization.
PieChart(document.getElementById('chart_div'));
    chart.draw(data, options);
}
</script>
26 </head>
<body>
<div id="chart_div" style="width:400; height:300"></
    div>
</body>
</html>

```

Oltre all'assenza di immagini, si segnala tra i motivi per la quale questa tecnologia è stata scartata, la totale assenza di supporto per mappe o coordinate geografiche.

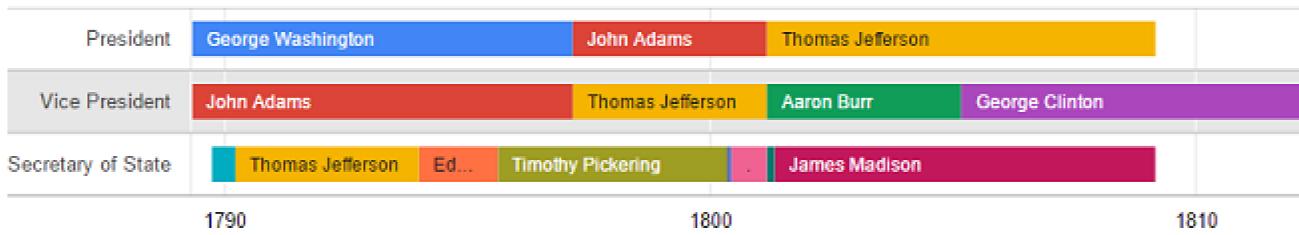


Figura 3.4: Screenshoot di una Timeline creata con Google Charts

3.3.5 Timemapper

Si tratta di un progetto open-source della Open Knowledge Foundation Labs [10], che si definisce come «una comunità di hacker civili e semplici cittadini incuriositi dalla possibilità di combinare tecnologia e le informazioni per rendere il governo più responsabile, la cultura più accessibile e la scienza più efficiente». Tale motto, in linea con la filosofia di ROUTE-TO-PA e di SPOD, e la forte preferenza per gli open data e i software gratuiti o open source ha subito attirato le nostre attenzioni.

Nasce come evoluzione di Timeline JS e ne eredita tutte le funzioni estendendo il supporto delle mappe: anzichè generare una mappa per ogni evento, Timemapper genera una singola mappa globale che racchiude tutte le coordinate geografiche degli eventi. Timemapper non utilizza balloon, i dati sono visualizzati sotto forma di diapositive che supportano testo e immagini. In questa visualizzazione l'utente può navigare tra i dati scorrendo le diapositive, cliccando sulle varie sezioni della Timeline o muovendosi tra i puntatori della Mappa. Le date possono rappresentare eventi singoli o intervalli.

Tra le componenti open-source di Timemapper troviamo Timeline JS, ReclineJS, Leaflet, Backbone e Bootstrap. Si tratta di un'applicazione Node.js orientata al Javascript, quest'ultimo ne gestisce buona parte della presentazione e visualizzazione.

I metadati sono conservati su Amazon S3 (Simple Storage Service, Servizio di archiviazione semplice) o filesystem locali, mentre per le date della Timeline si ricorre a Google Spreadsheet.

Non è attestata la possibilità di utilizzare JSON in alternativa al foglio di dati, motivo per cui si è deciso di escludere Timemapper e di ricorrere al "padre" di questa tecnologia, per non essere vincolati a Google Spreadsheet.

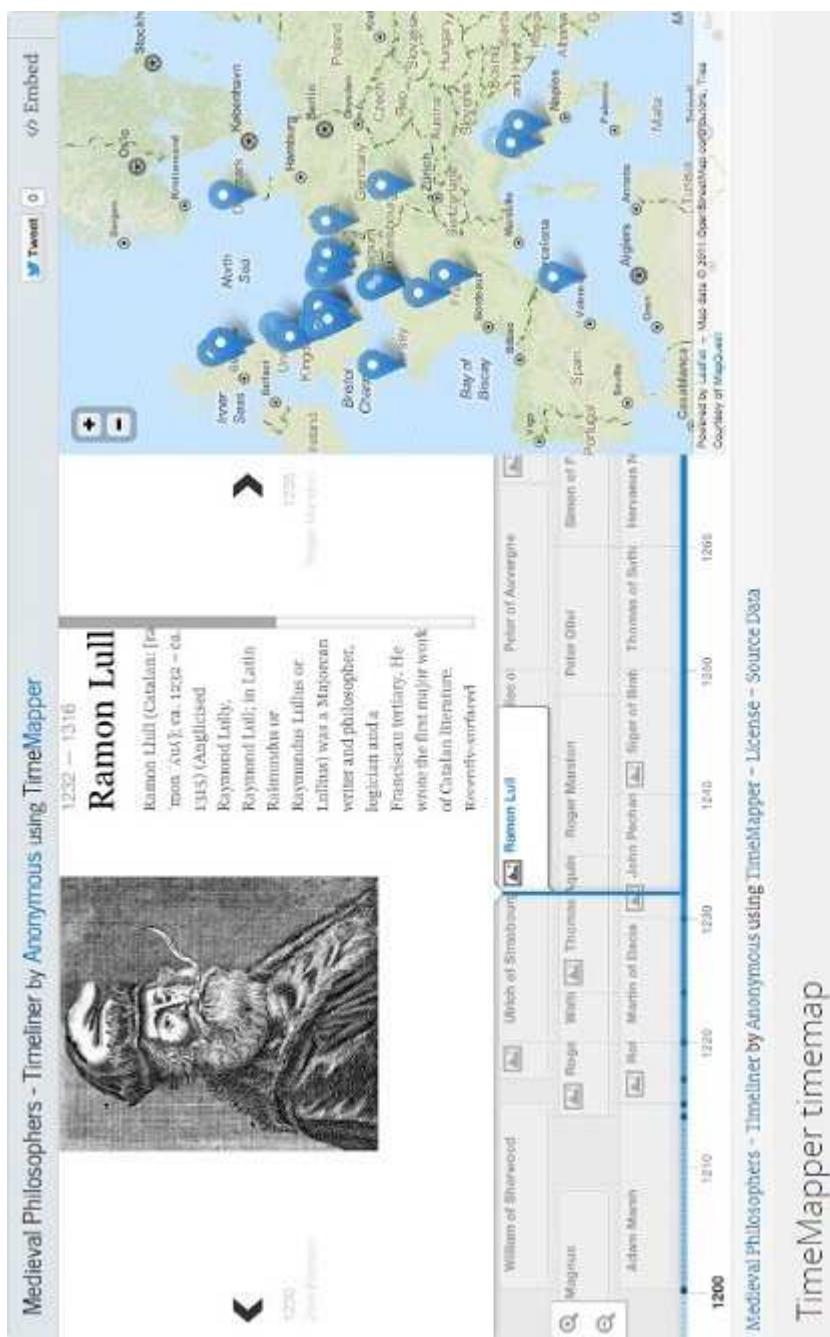


Figura 3.5: Screenshot di una Timeline con mappa creata con Timemapper

3.4 Tecnologie utilizzate

3.4.1 Javascript

La maggior parte dell'applicazione è scritta in Javascript, un linguaggio di scripting orientato agli oggetti e agli eventi per programmazione web lato client. JavaScript è stato standardizzato per la prima volta tra il 1997 e il 1999 dalla ECMA con il nome ECMAScript. L'ultimo standard, di giugno 2016, è ECMA-262 Edition 7. È anche uno standard ISO.

Il codice Javascript è inserito nel file HTML principale ("timeline-datalet.html", si veda il listato nell'Appendice). In particolare Javascript si occupa di convertire i dati ricevuti dalla datalet creator in formato JSON e di apportare eventuali modifiche su di essi, come convertire i numeri romani in intervalli di secoli, eliminare dati non conformi allo standard e operare sulle date incerte.

3.4.2 Polymer e Web Components

HTML (HyperText Markup Language, linguaggio a marcatori per ipertesti) è un linguaggio di markup nato per la formattazione e impaginazione di documenti ipertestuali. Si tratta di un linguaggio di pubblico dominio, la cui sintassi è stabilita dal World Wide Web Consortium (W3C) e la cui versione attuale, la quinta, è stata rilasciata nell'ottobre 2014.

Per estendere le funzionalità di HTML si utilizza la libreria Polymer, che consente la piena manipolazione delle Web Components. Con le Web Components è possibile creare elementi personalizzati riutilizzabili che interagiscono senza problemi con gli elementi built-in del browser, mantenendo il codice pulito e semplificandone la manutenzione.

Le Web Components [11] sono un set di caratteristiche attualmente prese in considerazione dal W3C per l'inclusione nelle specifiche dello standard HTML e della Document Object Model (DOM). Esse consentono la creazione di widgets riutilizzabili, o componenti, in documenti ed applicazioni web. Lo scopo del loro utilizzo è l'introduzione dell'ingegneria del software orientata a componenti nel World Wide Web. Il modello a componenti consente l'incapsulamento e l'interoperabilità di elementi HTML individuali. Si possono dividere in 4 tipologie principali: Elementi personalizzati (API per definire nuovi elementi HTML), Shadow DOM (incapsulano il DOM e i fogli di stile con la formattazione), HTML Imports (metodi dichiarativi per importare documenti HTML all'interno di altri documenti) e HTML Template (il tag <template> che consente di inserire pezzi inerti di DOM all'interno di documenti).

La libreria Polymer [12] fornisce una serie di caratteristiche progettate per semplificare e velocizzare la creazione di elementi personalizzati che si comportano come elementi standard del modello a oggetti del documento. Gli elementi di Polymer possono essere istanziati usando un costruttore o document.createElement, si possono configurare utilizzando attributi o proprietà e sono in grado di rispondere a cambiamenti di tali proprietà e attributi. Segue ora una tipica definizione di un oggetto Polymer:

```
<!DOCTYPE html>
<dom-module id="element-name">
  <template>
    <style>
      </style>
      5   <div>{{greeting}}</div>
    </template>
    <script>
      Polymer({
        10      is: "element-name",
        properties: {
          greeting: {
            type: String,
            value: "Hello!"
        }
        15      }
      });
    </script>
</dom-module>
```

3.4.3 JSON

JSON, acronimo di JavaScript Object Notation [13], è un formato di scambio dati basato su coppie nome-valore. Esso risulta di facile lettura sia per un essere umano che per una macchina, ed è compatibile con virtualmente ogni linguaggio di programmazione. Si basa su un sottoinsieme del Linguaggio di Programmazione JavaScript, Standard ECMA-262 Terza Edizione - Dicembre 1999. In diversi linguaggi, le coppie nome-valore vengono realizzate come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo. Inoltre JSON supporta elenchi ordinati di valori, comunemente implementati come array, vettori, elenchi o sequenze.

Proprio l'universalità di tali strutture dati consente la completa integrazione di JSON o il pieno interscambio di dati attraverso questo formato tra i diversi linguaggi di programmazione che lo supportano.

In JSON un oggetto è una serie non ordinata di coppie nomi-valori che inizia con una parentesi graffa sinistra e finisce con una parentesi graffa destra. Ogni nome è seguito da due punti (:) ed ogni coppia di nome-valore è separata da una virgola.

Un array invece è una raccolta ordinata di valori che comincia con una parentesi quadra sinistra e finisce con una parentesi quadra destra. I valori sono separati da una virgola.

Javascript dispone di parser per il linguaggio in entrambe le direzioni: è possibile tradurre oggetti JSON in oggetti Javascript e viceversa. Si è scelto di utilizzare questo formato per l'interscambiabilità fornita naturalmente e poichè diverse datalet già supportano l'utilizzo di JSON come formato di scambio dati.

3.5 Timeline JS3



Figura 3.6: Logo di Knightlab



Figura 3.7: Logo di Timeline JS

Dopo un attento lavoro di ricerca e dopo aver esaminato le tecnologie disponibili, il lavoro si è orientato verso un'integrazione di Timeline JS3 [14] nel plug-in "DEEP" di SPOD. Si tratta di un tool open source sviluppato dalla Northwestern University tramite il suo team Knight La che permette di realizzare Timeline ricche e interattive. Utilizzato anche dai quotidiani Time, Le Monde e CNN, esso è disponibile in più di 60 lingue. Anziché utilizzare i balloon come in buona parte delle tecnologie precedenti, i dati vengono mostrati agli utenti all'interno di diapositive contenenti testi o contenuti multimediali. È possibile navigare come in una presentazione o cliccando direttamente sulla Timeline e dispone di meccanismi di Zoom. Consente di incorporare mappe tramite Google Maps per mostrare le coordinate geografiche di un evento e supporta mappe stradali, satellitari, ibride e geologiche. Oltre alle mappe, è possibile inserire immagini, video, clip audio o pagine da wikipedia. Richiede Google spreadsheet per utenti meno esperti, mentre gli utenti esperti possono avvalersi di JSON e Javascript per avere maggiore

controllo sui dati inseriti durante la creazione della Timeline. Per integrare il tool in una pagina web bastano un tag link per caricarne i CSS e un tag script per caricare la libreria stessa. Un altro script si occupa di creare la Timeline tramite Spreadsheet (il cui template è fornito dal tool stesso) o JSON (a cura del programmatore).

Timeline JS3 è sottoposto a Mozilla Public License (MPL) [15] che consente di produrre un "opera più ampia" utilizzando materiale sotto altre license, tra cui la MIT in uso su SPOD. Eventuali modifiche al codice originariamente sotto licenza MPL devono rimanere sotto tale licenza, ma è consentito all'autore di aggiungere il proprio nome al termine della lista degli sviluppatori del software.

3.6 Confronto tra tecnologie

Prima di procedere oltre con l'analisi in dettaglio del codice utilizzato per integrare Timeline JS3 in DEEP, viene ora proposta una tabella di riepilogo 3.1 che analizza le caratteristiche comuni e le differenze tra le 6 diverse tecnologie analizzate fin'ora ed una che confronta le ultime release stabili 3.2 e gli ultimi commit su GitHub per ogni singola tecnologia:

Tabella 3.1: Confronto tra le 6 tecnologie analizzate

Nome	Immagini	Intervalli	Zoom	Balloon	Geolocalizzazione
Chronozoom	Si	Si	Si	Si	Solo per generare sotto-sezioni
Crossfilter	No	Si	No	No	No
Timeline di Simile	Si	Si	No	Si	No
Google Charts	No	Si	No	Si	No
Timemapper	Si	Si	Si	Diapositive	Si, obbligatorio
Timeline JS3	Si	Si	Si	Diapositive	Si, tramite Google Maps

Tabella 3.2: Ultime release stabili ed ultimi commit su GitHub. Dati aggiornati al 25 Novembre 2016

Nome	Ultima release su GitHub	Ultimo commit su GitHub
Chronozoom	Beta 2.0 14, Marzo 2012	14 Agosto 2015
Crossfilter	9 Giugno 2016 (1.3.14)	16 Settembre 2016
Timeline di Simile	16 Novembre 2013 (1.0)	25 Luglio 2016
Google Charts	16 Novembre 2013 (1.0)	19 Novembre 2016
Timemapper	6 Settembre 2011 (flask-v0.1)	15 Luglio 2016
Timeline JS3	15 Novembre 2016 (3.4.3)	23 Novembre 2016

3.7 Codice in dettaglio

3.7.1 Manipolazione di date

La prima difficoltà con cui ci si è dovuti scontrare è stata l'assenza di un formato unico per la datazione nei vari dataset providers. Non esistendo una convenzione con cui indicare le date incerte, nè un modo per rappresentarle all'interno di Timeline JS, si è optato per impostare a 0 le date di eventi completamente incerti per poi nasconderli dal risultato finale, mentre per eventi la cui datazione corrisponde ad un intervallo dubbio si è optato per trasformare l'evento in un intervallo; ad esempio un castello la cui data di costruzione è compresa tra il 1750 e il 1800, avrà come data di inizio il 1750 e come data di fine il 1800, mentre un manufatto che non è stato possibile datare non verrà mostrato sulla timeline. La decisione di nascondere oggetti con date nulle deriva dalla definizione stessa di Timeline citata all'inizio del capitolo: "modo di rappresentare visivamente eventi in ordine cronologico"; è evidente che sovraccaricare la data 0 di eventi a datazione incerta è controproducente per l'utente, che si trova a navigare con maggiore difficoltà nella Datalet, oltre che ambiguo nel caso in cui esistano eventi certi realmente accaduti nell'anno 0. Per il codice si faccia riferimento all'appendice, listato "timeline-datalet" A.2.1. Le funzioni trattate in questa sezione comprendono date(date), about(date), beforechrist(date), numex(txt) e cleandata(data, todelet).

La funzione date riceve come primo parametro l'array di date su cui lavorare e come secondo parametro un valore booleano: se questo è impostato su "false" (falso), alcune parti di codice verranno saltate. Se tale parametro è impostato su "true" (vero), all'inizio della funzione viene creato un array di valori booleani di grandezza pari al numero di eventi, questi valori booleani sono inizialmente impostati su "false". Ogni volta che una data viene impostata a 0, se il parametro booleano della funzione è "true" si imposta la cella dell'array di booleani corrispondente all'indice dell'evento a "true".

La funzione che si occupa di operare le trasformazioni necessarie alle date prima di convertirle in formato JSON per passarle in input a Timeline JS, effettua un controllo per ricercare le stringhe '?' (punto interrogativo) e '' (stringa vuota), se la ricerca ha successo, tali date sono impostate a 0. La ricerca specifica di queste due stringhe è stata influenzata dalla pratica comune su SPOD di utilizzare tali stringhe per indicare date incerte.

Una volta esclusa la presenza delle due stringhe si controlla se il formato dei dati passati in input è di tipo 'number' (numerico), in caso di controllo positivo non occorrerà effettuare alcun tipo di trasformazione sui dati.

Qual'ora il controllo precedente abbia dato esito negativo, si verifica che il tipo di dati passato in input sia una stringa: in caso di esito negativo la data verrà impostata a 0 e si procederà al dato successivo (si assume che ci si trovi davanti a dati corrotti o nulli). Nel caso in cui ci si trovi di fronte ad una stringa, il programma avvierà una serie di passi per trasformare la stringa in un intero.

Il primo passo consiste nell'utilizzo di un'espressione regolare per verificare che all'interno della stringa siano presenti numeri arabi: potrebbe trattarsi infatti di numeri mescolati ad annotazioni personali dell'utente, come "1700 circa", "fine 1800" o "1500-1600". In questo caso, la funzione `about(date)` stabilirà se sia presente nella stringa una delle 4 espressioni che sono state selezionate per indicare incertezza: "circa", "intorno", "?" o "ca": se restituisce un esito positivo, alla descrizione dell'evento viene incluso il messaggio "Data incerta" colorato di rosso per avvisare l'utente che la data su cui è stato posizionato l'evento è un'approssimazione.

A questo punto si verificherà la presenza specifica del carattere "-" (trattino alto) utilizzato comunemente per definire intervalli: qual'ora venga trovato, si procederà a dividere la stringa in due sotto-stringhe tramite la funzione `split()` a cui verrà dato in input proprio "-". La data verrà ora trattata come un intervallo e si chiamerà ricorsivamente la funzione `date()` per operare sulle due date estratte ma il valore booleano sarà impostato su "false".

Prima di procedere ulteriormente nella descrizione della funzione, occorre fare un breve digressione sui diversi formati della datazione: le date si dividono in date successive alla nascita di Cristo e date antecedenti la nascita di Cristo. La nascita di Cristo, il cosiddetto anno 0, pur essendo incerta, viene collocata a circa 753 anni dopo la fondazione di Roma o, se si preferisce, 776 anni dopo la prima Olimpiade. Per convenzione le date relative ad eventi avvenuti dopo la nascita di Cristo vengono indicate con i soli numeri arabi, anche se per indicare tali eventi esistono le locuzioni "Dopo Cristo" (abbreviata in DC), "Anno Domini" (AD) o le espressioni inglesi "Common Era" (CE) e "After Christ" (AC). Per esempio, dire "nell'anno 2017" corrisponde all'espressione "nell'anno 2017 Dopo Cristo". Per le date antecedenti l'anno 0 esistono diverse convenzioni: si utilizzano numeri negativi, si inserisce la sigla AC ("Avanti Cristo") o il suo equivalente inglese "BC" ("Before Christ") o si utilizza l'espressione "before the Common Era" (BCE). Di conseguenza, la fondazione di Roma andrebbe indicata con "753 Avanti Cristo", o abbreviata "753 A. C.".

Timeline JS richiede che gli eventi antecedenti l'anno 0 abbiano come data di inizio un numero negativo (ad esempio -753), motivo per cui nel caso dell'utilizzo di locuzioni si è reso necessario una ricerca di tali espressioni nella

stringa contenente la data. Si è assunto, per evitare fraintendimenti dovuti all'espressione "AC" che può significare "Avanti Cristo" (numero negativo) o "After Christ" (positivo) di assumere come convenzione l'utilizzo delle locuzioni solo per le date precedenti l'anno 0. Di conseguenza, l'espressione "1800" indicherà l'anno 1800 Dopo Cristo, mentre le espressioni "1800 a.c." o "1800 BC" verranno assunte come l'espressione "1800 anni prima della nascita di cristo" (e di conseguenza convertite in -1800). La funzione date() chiama la funzione isbeforechrist() su un numero per verificare se ci sia bisogno di convertirlo: questa restituisce True nel caso in cui siano presenti le locuzioni "a.c.", "ac", "bc" o "b.c.". Non essendo la funzione includes() "case sensitive", per evitare di effettuare un controllo su tutte le possibili combinazioni di lettere maiuscole e minuscole la stringa viene prima trasformata nella sua equivalente costituita da sole lettere minuscole tramite la funzione toLowerCase(). Il controllo sulle stringhe "bc" e "b.c." garantisce un riscontro positivo anche qual'ora si utilizzassero le espressioni "bce" e "b.c.e." in quanto tali espressioni le contengono a loro volta. Una volta estratto il numero, per convertirlo basta moltiplicarlo per -1.

L'estrazione del numero dalla stringa viene effettuata dalla funzione numex(txt) che chiama prima la funzione match() dando in input l'espressione regolare che corrisponde ad un numero arabo, ed in seguito la funzione join() per poi restituire l'intero estratto.

Nel caso in cui la ricerca di numeri arabi non sia andata a buon fine, si effettua un'ulteriore ricerca, stavolta per i numeri romani (I, II, III, IV, ...) tramite la funzione romanize(string). Se anche quest'ultima non dovesse avere successo, la funzione provvederà ad assegnare alla data il valore 0 e procederà con la data successiva.

Terminata anche la ricerca di numeri romani, se il parametro booleano della funzione era stato impostato a "true", viene chiamata la funzione cleandata() che si occupa di annullare i parametri media, nome e descrizione degli eventi la cui data sia stata forzata a 0 localizzandoli tramite le celle dell'array "todelet": se la cella "i" è impostata a "true", vuol dire che bisogna annullare le celle i degli array data[1].data (nome), data[2].data (descrizione) e data[3].data (media), in questo modo la funzione makethejson() li ignorerà nella costruzione del JSON e non verranno visualizzati sulla Timeline.

3.7.2 Manipolazione di numeri romani

Esamineremo ora in dettaglio le operazioni che vengono effettuate per identificare e localizzare numeri romani. Le funzioni trattate in questa sezione comprendono romanize(string), fromRoman(string) e romaninterval(number).A.2.1 Il sistema di numerazione romano [16] è un sistema di numerazione additivo in cui ad ogni simbolo è associato un valore e il numero rappresentato è dato dalla somma dei valori dei "simboli letterari" che lo compongono. Il sistema utilizza 7 simboli a cui è assegnato un valore numerico: la lettera I indica il numero 1, la V il 5, la X il 10, la L il 50, la C il 100, la D il 500 e la M il numero 1000. Esistono alcune semplici regole per combinarli tra di loro, che verranno ora di seguito elencate per una maggior comprensione dell'algoritmo di cui discuteremo in questa sezione. Innanzitutto all'interno di un numero romano i simboli I, X, C e M possono essere ripetuti consecutivamente al massimo tre volte, mentre i simboli V, L e D non possono essere mai inseriti più di una volta consecutiva; una stringa di simboli che non presenta mai valori crescenti denota l'intero ottenuto sommando i valori dei simboli indicati, mentre quando si incontra un simbolo seguito da un secondo simbolo di valore maggiore si ha come risultato la differenza tra i due (solo I, X e C possono essere usati a tale scopo). Per fare un esempio, la stringa III indica il numero 3 ($1+1+1$), mentre per indicare il numero 4 si userà la stringa IV (ovvero 5-1). Sono accettabili anche stringhe formate da coppie del tipo precedente e simboli, purché si passi da una coppia a una coppia di valore inferiore, da un simbolo a una coppia di simboli entrambi inferiori e da una coppia a un simbolo inferiore di entrambi i membri della coppia. Tuttavia il sistema comunemente conosciuto non è quello utilizzato nell'antica Roma, ma una versione modificata nel periodo Medioevale poichè l'originale risultava troppo lungo per descrivere alcuni numeri. Il sistema originale prevedeva che i valori dei simboli venissero solo addizionati e mai sottratti.

Nella storiografia i numeri romani vengono utilizzati per rappresentare secoli, con il numero romano I che indica il primo secolo, la cui durata comprende gli anni da 0 a 99, il II che indica il secondo secolo (100-199) e così via. Di conseguenza quando verrà incontrato un numero romano lo si tratterà come un intervallo e non come una data singola. Ad esempio, una volta trovata la stringa XVIII, che corrisponderebbe al numero 18, il sistema dovrà prima convertirlo nel suo equivalente, sottrargli 1 e a quel punto moltiplicarlo per 100 e generare l'intervallo corrispondente (nel caso in esempio 1700-1799). La funzione che si occupa della maggior parte del lavoro sui numeri romani è romanize(), che viene invocata dalla funzione date() nel caso in cui i con-

trolli effettuati sulla presenza di numeri arabi o stringhe indicati una data ignota falliscano. All'inizio della funzione viene definita un'espressione regolare che identifica il tipico pattern descritto dalle regole per scrivere un numero romano, la stringa passata in input viene divisa in sotto-stringhe tramite la funzione `split()` e viene creato un array vuoto che avrà il compito di contenere tutti i numeri romani convertiti che si andranno a trovare. La scelta di utilizzare un array anzichè un singolo numero è stata influenzata dalla pratica comune sui datalet contenuti su SPOD di inserire due numeri romani in una singola data per definire un intervallo. La variabile temporanea "half" viene utilizzata per sommare 50 anni, qual'ora sia necessario, al numero appena ottenuto. La decisione di sommare 50 anni viene prese dall'`if` successivo, che esamina se una delle sotto-stringhe della sequenza originaria contenga la stringa "second", poiché a volte viene utilizzata l'espressione "second half of", o l'equivalente italiano "seconda metà del", per indicare che l'evento si svolge negli ultimi 50 anni del secolo preso in esame.

Tramite la funzione `match` si controlla se la stringa rispetta le regole dei numeri romani, in caso di controllo positivo viene invocata la funzione `from-Roman(string)`, altrimenti si procede alla prossima sottostringa.

La funzione che converte i numeri romani nel loro equivalente arabo, prende in considerazione solo i primi tre simboli della numerazione romana: I, V ed X. Avendo deciso di considerare i numeri romani secoli che verranno utilizzati per indicare eventi avvenuti dopo l'anno 0, ed essendo il secolo corrente indicato con la stringa XXI, appare chiaro che l'utilizzo del simbolo L (per indicare il secolo XL, ovvero a partire dal 3900 dopo cristo) o degli altri 3 simboli non presi in considerazione non avverrà per un lasso di tempo decisamente lungo per la storia umana.

Nel primo ciclo `for` vengono inseriti i valori delle singole lettere in un array temporaneo, senza tenere conto della loro posizione. Prendendo ad esempio il numero XIX (19), al termine del primo `for` l'array conterrà i numeri 10, 1 e 10. A questo punto il secondo ciclo `for` somma i numeri contenuti nell'array temporaneo seguendo le regole della numerazione romana: se il numero `i` preso in esame è maggiore o uguale del numero `i+1`, verrà sommato normalmente, altrimenti verrà sommato il numero `i+1` e verrà sottratto il numero `i`, incrementando il valore di `i` di 2 anzichè di 1. Nel numero preso in esempio, alla prima iterazione si otterrebbe 0+10; alla seconda si otterrebbe 10+10-1. Una volta che il valore viene restituito alla funzione `date()`, questo viene moltiplicato per 100 e gli viene sottratto 100, per poi sommarlo con il valore della variabile `half` (che può assumere valore 0 o 50, in entrambi i casi essa viene azzerata dopo la somma).

Terminata l'iterazione sulle sottostringhe, se l'array da restituire risulta vu-

to poichè non sono stati trovati numeri romani vengono inseriti due 0 per rendere incerta la data nel caso in cui sia stata passata una stringa contenente solo parole o commenti dell'autore del datalet. Questi valori indicheranno alla funzione date() di impostare la cella dell'array "todelet" corrispondente a questo numero a "true".

Nella funzione date(), i due numeri restituiti da romanize() vengono usati per creare l'intervallo corrispondente, altrimenti se viene restituito un solo numero dalla funzione romanize() viene generato un intervallo avvalendosi della funzione romaninterval(number): questa restituisce il numero passato in input dopo avergli sommato 49 (se il resto della divisione per 100 è 50) o 99 (se il resto è 0), in questo modo è possibile ottenere il secolo a cui si riferisce l'evento.

3.7.3 La conversione dei dati in formato JSON

Dopo aver manipolato le date di inizio e, se presenti, quelle di fine, bisogna convertire i dati in formato JSON. Questo lavoro viene svolto dalla funzione makethejson(events) e da una serie di funzioni di supporto che saranno discusse in questa sezione. Per il codice si faccia riferimento al listato nell'appendice A.2.1

Knightlab fornisce su GitHub un esempio di come formattare JSON affinchè Timeline JS possa leggerli senza problemi. [17] Questo schema è stato seguito nella conversione da oggetti a JSON. In particolare distinguiamo una gerarchia di tipi in cui il singolo oggetto JSON ha come figlio un array "events" che al suo interno contiene i singoli eventi. I singoli eventi contengono a loro volta oggetti media o mappa, oggetti di tipo text e oggetti di tipo date. Questi ultimi contengono dei campi che costituiscono le informazioni più semplici disponibili: i media e le mappe contengono una "url", un "caption" ed un "credit"; text contiene un campo "headline" ed un campo "text"; date contiene i campi "year", "month" e "day" anche se al momento solo il campo "year" è utilizzato.

La funzione makethejson() inizializza un oggetto json vuoto e crea un'espressione regolare che sarà utilizzata per distinguere tra gli oggetti media e gli oggetti mappa, questa espressione definisce il pattern tipico di una url. A questo punto crea un campo "events" all'interno dell'oggetto JSON vuoto appena creato: questo sarà il campo che verrà popolato dalla funzione.

Il primo controllo che viene effettuato stabilisce il tipo di media: innanzitutto viene effettuato un controllo per verificare che l'evento in questione abbia effettivamente un media; nel caso in cui il controllo dia esito negativo (campo vuoto o impostato su null) si procede al controllo successivo, altrimenti si inizia a costruire l'oggetto media. Se è presente un "caption" (selezionato dalla controllet tramite il primo campo) viene inserito, altrimenti lo si lascia vuoto. Tramite l'espressione regolare si verifica se il campo 4 della controllet contenga una URL: in caso di riscontro positivo avremo davanti un oggetto media e inseriremo la URL nel campo "url", specificando che l'oggetto è di tipo media. In caso contrario si tratterà di coordinate, e le inseriremo al posto dell'url specificando che l'oggetto è di tipo "map". Queste informazioni verranno utilizzate in seguito dalla funzione makemedia() chiamata da makeevent(). Discuteremo in seguito di queste funzioni.

Si costruisce ora un oggetto "data" con il nome di "start_data", prelevando l'anno manipolato in precedenza (è passato come campo 5 dalla controllet). Se la data è di tipo array (perchè si tratta di un secolo specificato in numeri romani o di un intervallo che l'utente ha erroneamente inserito in un

solo campo) viene creato anche un secondo oggetto "data" con il nome di "end_date". Se il sesto campo della controllet è stato compilato, questo viene utilizzato per costruire eventuali date di fine degli eventi, con il nome di "end_date" (queste hanno la priorità sulle "end_date" create al passo precedente). Vengono effettuati dei controlli di coerenza sui dati per evitare che la data di fine sia precedente alla data di inizio.

Termina la creazione dell'oggetto date, viene costruito l'ultimo oggetto necessario alla creazione dell'evento, ovvero un oggetto "text" che incorpora come elemento "caption" il valore passato tramite il secondo campo della controllet e come "text" il terzo campo.

Prima di inserire l'oggetto vengono eliminati eventuali dati corrotti o quelli privi di data identificati in precedenza: se un oggetto non ha contemporaneamente il "text", il "caption" ed un "media" viene scartato. Se invece l'oggetto supera il controllo viene invocata la funzione makeevent(media, start_date, end_date, text).

Questa funzione crea un oggetto json event vuoto, e poi lo popola con i dati passati in input. Se il primo parametro è di tipo "media" ed è diverso da null viene invocata la funzione makemedia(url, caption, credit) che i occupa di formattare in formato JSON e inserire nell'oggetto event i dati passati in input. Se invece è di tipo "url" viene invocata la funzione makemap(latlong, caption, credit) che genera una url di Google Maps tramite il parametro "latlong" che è una stringa contenente la latitudine e la longitudine dell'evento per poi invocare nuovamente makemedia() passando la url appena creata come parametro url.

Il secondo parametro contiene la data di inizio dell'evento, che viene passata come parametro alla funzione makedate(year, month, day).

Il terzo parametro è opzionale, e contiene il testo associato all'evento composto da un titolo e da una descrizione: vengono elaborati dalla funzione maketext(headline, textparam).

Anche il quarto parametro è opzionale, e contiene l'eventuale data di fine dell'evento (utile per gli intervalli).

Dopo aver iterato su tutti gli eventi, la timeline è completa e verrà generata dall'ultima istruzione di presentData(): new TL.Timeline().

Capitolo 4

Conclusioni

Si é realizzata una nuova visualizzazione per Open Data, integrandola su SPOD e all'interno di DEEP senza modificarne l'architettura principale, permettendo agli utenti di visualizzare i dati fortemente legati al tempo in un formato interattivo e di facile consultazione. Questa Datalet trova ampio utilizzo nel campo dei beni culturali e dell'archeologia, in particolare oltre che per visualizzare eventi, anche per le datazioni di reperti archeologici o di monumenti storici che prima venivano mostra su SPOD in una semplice visualizzazione su tabelle.

La fase di creazione guidata ricalca quella delle altre datalet, rendendo il processo molto semplice per l'utente comune di SPOD.

Si é attuato inoltre un sistema di controllo degli errori all'interno del dataset prima della sua trasformazione in Timeline, in questo modo finché tutti i dataset non saranno standardizzati, sarà possibile visualizzare almeno le parti dei dataset attualmente conformi allo standard all'interno della datalet, senza pregiudicare la visualizzazione della Timeline.

Per quanto riguarda gli sviluppi futuri, attualmente il sistema non é in grado di distinguere una colonna di dati contenente anni da una contenente secoli, salvo i casi in cui i secoli siano indicati con il sistema di numerazione romano.

L'intenzione é di rendere questa datazione uno standard per gli Open Data in futuro. Nel caso in cui non fosse possibile ridefinire lo standard, occorrerà modificare il sistema, aggiungendo un campo opzionale "Secoli" alternativo al campo "Data d'inizio" in cui i valori vengano interpretati a prescindere come intervalli e non come date singole. Un'ulteriore alternativa, poco pratica date le diverse lingue in uso su SPOD, sarebbe quella di basare la conversione degli anni sul nome della colonna a cui appartengono nel dataset

originale.

Risultano interessanti le possibilità offerte da Timemapper, che dimostra come sia possibile realizzare grazie a Timeline JS3 una mappa globale interattiva con puntatori (map-marker) allegata alla Timeline, anziché avere singole mappe prive di puntatori per ogni singolo evento. Sfortunatamente l'integrazione di Timemapper non è semplice come quella della tecnologia "padre" Timeline JS, in quanto non esiste ufficialmente un supporto per il JSON: i dati infatti vengono prelevati esclusivamente da Google Spreadsheet, anche se online sono presenti diverse soluzioni per permettere a Timemapper di accettare i dati in formato JSON [18].

Inoltre non esiste una versione libreria di Timemapper (timemapper.js), il che impedisce di includerlo in altri progetti così come accade per Timeline JS3 [19]. Non ci sono piani da parte di Open Knowledge Foundation Labs di una release in formato libreria, anche se sono presenti delle soluzioni per aggirare il problema su GitHub [20]. Si propone dunque di adottare le soluzioni disponibili in rete per adattare Timemapper a DEEP e di integrarlo nella Timeline descritta in questo lavoro di tesi.

Bibliografia

- [1] ROUTE-TO-PA: <http://routetopa.eu/> 1
- [2] SPOD: <http://spod.routetopa.eu/> 1
- [3] Definizione di Open Data: https://it.wikipedia.org/wiki/Dati_aperti 2.1
- [4] Pagina per Sviluppatori di OXWALL: <https://developers.oxwall.com/> 2.3
- [5] DEEP su GitHub: <https://github.com/routetopa/deep> 2.4
- [6] Chronozoom: <http://www.chronozoom.com/> 3.3.1
- [7] Crossfilter: <http://square.github.io/crossfilter/> 3.3.2
- [8] Timeline di Simile Widgets: <http://www.simile-widgets.org/wiki/Timeline> 3.3.3
- [9] Google Charts: <https://developers.google.com/chart/interactive/docs/gallery/timeline> 3.3.4
- [10] Timemapper: <http://timemapper.okfnlabs.org/> 3.3.5
- [11] Definizione di Web Components: https://en.wikipedia.org/wiki/Web_Components 3.4.2
- [12] Polymer: <https://www.polymer-project.org/1.0/> 3.4.2
- [13] JSON: <http://www.json.org/json-it.html> 3.4.3
- [14] Timeline JS3: <https://timeline.knightlab.com/> 3.5
- [15] Mozilla Public License 2.0: <https://www.mozilla.org/en-US/MPL/2.0/> 3.5

- [16] Sistema di numerazione Romano: https://it.wikipedia.org/wiki/Sistema_di_numerazione_romano 3.7.2
- [17] Timeline su Whitney Houston in JSON per Timeline JS3: <https://github.com/NUKnightLab/TimelineJS3/blob/master/website/templates/examples/houston/timeline3.json> 3.7.3
- [18] Data sources other than google docs spreadsheet such as raw CSV or JSON files: <https://github.com/okfn/timemapper/issues/1074>
- [19] Is it possible to use timemapper as a simple js library? <https://github.com/okfn/timemapper/issues/112> 4
- [20] Factor out core JS library as timemapper.js so others can use in their own apps easily: <https://github.com/okfn/timemapper/issues/73>

Appendice A

Appendice

A.1 Immagini

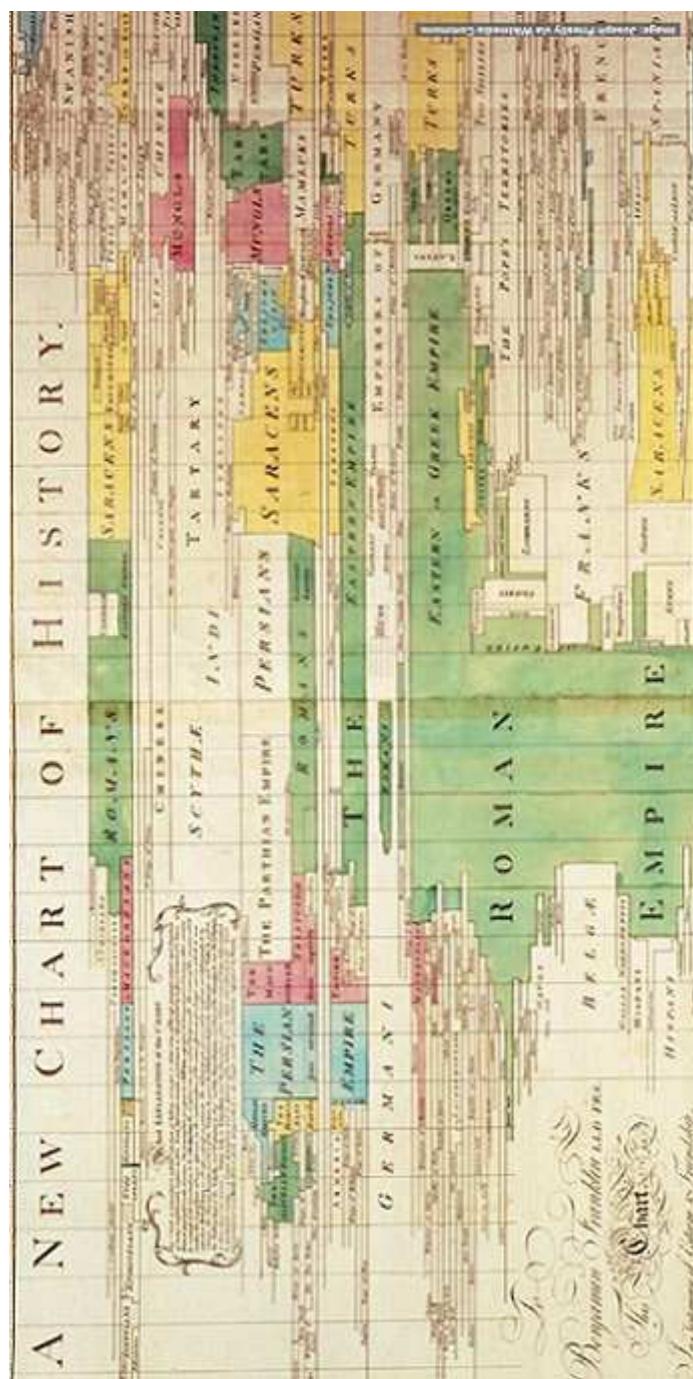


Figura A.1: "A New Chart of History" di Joseph Priestley (1765)



Figura A.2: Screenshot della demo allegata a Timeline Datalet



Figura A.3: Creazione di una Timeline Datalet. Fase 1: Selezione della visualizzazione



Figura A.4: Creazione di una Timeline Datalet. Fase 2: Inserimento dei dati nei campi

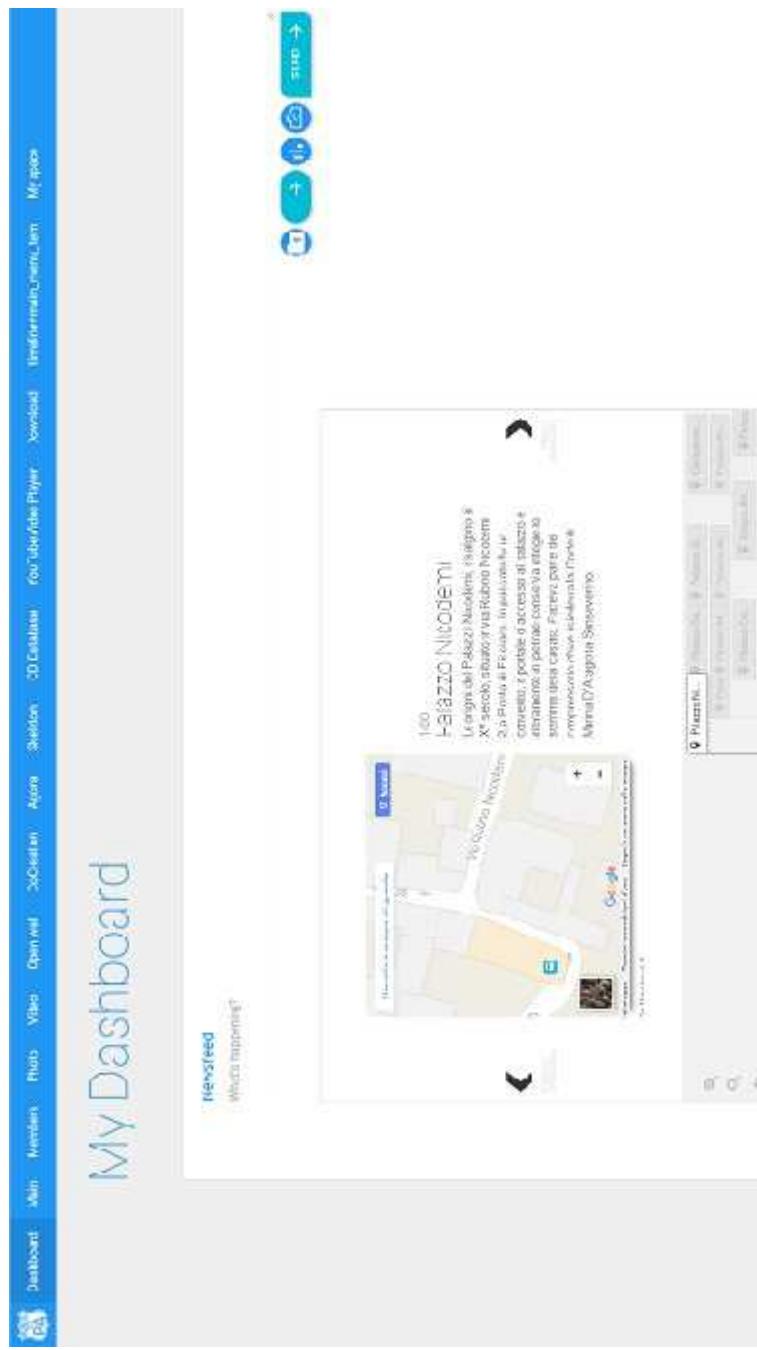


Figura A.5: Creazione di una Timeline Datalet. Fase 3: La Datalet nella Bacheca di SPOD

A.2 Listati

A.2.1 timeline-datalet

```

1 <link rel="import" href="../base-ajax-json-alasql-
    datalet/base-ajax-json-alasql-datalet.html">
2   <link title="timeline-styles" rel="stylesheet" href="-
    https://cdn.knightlab.com/libs/timeline3/latest/css-
    /timeline.css">
3   <script src="https://cdn.knightlab.com/libs/timeline3/
    latest/js/timeline.js"></script>
4
5     <dom-module id="timeline-datalet">
6       <template>
7
8         <div id='timeline_embed' style="width:100%;-
    height:600px"></div>
9           <base-ajax-json-alasql-datalet data-url="{
10             dataUrl}" fields="{{fields}}" data="{{data}}"
11             title="{{title}}" description="{{description}}"
12             export-menu="{{export-menu}}"></base-ajax-json-
13             alasql-datalet>
14           </template>
15
16         <script>
17           var TimelineBehavior = {
18
19             presentData: function() {
20
21               if (!this.data) return;
22
23                 this.data[4].data = this.date(this.
24 data[4].data, true);
25
26               if (this.data[5] != null) {
27                 this.data[5].data = this.date(this
28 .data[5].data, false);
29               }
30
31             }
32
33           </script>
34         </dom-module>
35       </template>
36     </dom-module>
37   </script>
38 </body>
39 </html>
```

```

26           var timelinejson = this.makethejson(
this.data);
            this._component.timeline = new TL.
Timeline(this._component.$._timeline_embed,
timelinejson);
        },

31         makethejson: function (events) {
            var item = {};
            var regex = new RegExp("[ --zA-Z0-9@:_%.\u00a0\u202f]+#]{2,256}\u00a0.[a-z]{2,6}\u00a0b([-a-zA-Z0-9@:_%.\u00a0\u202f]+#?&//=]*");
            var media = "";
            var caption = "";
            var i = 0;
            item["events"] = [];
36            while (i < events[0].data.length){
                if (events[3].data[i]==="" || events
[3].data[i]==="null" || events[3].data[i]==null)
                    media = null;
                else{
                    if (events[0].data[i]!=="null")
                        caption = events[0].data[i]
41                ];
                    else
                        caption = "";
                    if (events[3].data[i]==null)
                        media = null;
                    else if (events[3].data[i].
match(regex))
                        media = [events[3].data[i
], caption, null, "media"];
                    else
                        media = [events[3].data[i
], caption, null, "map"];
                }
46
51            var start = [];
            var end = [];

```

```

        if( events [ 4 ] . data [ i ] . constructor
56      == Array ){
            start = [ events [ 4 ] . data [ i ] [ 0 ] ,
            null , null ];
            if( events [ 4 ] . data [ i ] [ 0 ] < events
                [ 4 ] . data [ i ] [ 1 ] )
                    end = [ events [ 4 ] . data [ i
                ] [ 1 ] , null , null ];
                } else {
                    start = [ events [ 4 ] . data [ i ] ,
                null , null ];
                    if( events [ 5 ] != null && events
                [ 5 ] . data [ i ] >= events [ 4 ] . data [ i ] )
                    end = [ events [ 5 ] . data [ i ] ,
                null , null ];
                    else
                        end = null ;
                }
                var text = [ events [ 1 ] . data [ i ] ,
            events [ 2 ] . data [ i ] ];
                if( media==null && text [ 0 ] == " " &&
            text [ 1 ] == " " ) {
                    } else
                        item [ " events" ] . push ( this .
makeevent ( media , start , end , text ) );
                        i = i + 1;
                }
                return item ;
            },
71
76      makemap: function ( latlong , caption ,
credit ) {
            var url = " https://www.google.com/maps
@"+latlong+",19z/";
            return this . makemedia ( url , caption ,
credit );
        },
        makemedia: function ( url , caption , credit )
{

```

```

81         var media = {};
82         media[ "url" ] = url;
83         if( caption!=null)
84             media[ "caption" ] = caption;
85         if( credit!=null)
86             media[ "credit" ] = credit;
87         return media;
88     },
89
90     makeevent: function (media, start_date,
91                           end_date, text) {
92         var event = {};
93         if( media!=null){
94             if( media[3]==="media")
95                 event[ "media" ] = this .
96                 makemedia( media[0], media[1], media[2]);
97             else if( media[3]==="map")
98                 event[ "media" ] = this .makemap(
99                     media[0], media[1], media[2]);
100            }
101           event[ "start_date" ] = this .makedate(
102               start_date[0], start_date[1], start_date[2]);
103           if( end_date!=null)
104               event[ "end_date" ] = this .makedate(
105                   end_date[0], end_date[1], end_date[2]);
106           if( text!=null)
107               event[ "text" ] = this .maketext( text
108                   [0], text[1]);
109           return event;
110     },
111
112     maketext: function (headline, textparam) {
113         var text = {};
114         if( headline!=null)
115             text[ "headline" ] = headline;
116         if( textparam!=null)
117             text[ "text" ] = textparam;
118         return text;
119     },

```

```

makedate: function (year , month , day) {
    var date = {};
    if(month!=null)
        date [ "month" ] = month;
    if(day!=null)
        date [ "day" ] = day;
    date [ "year" ] = year;
    return date;
},
116

numex: function (txt){
    var numb = txt.match(/\d/g);
    126    numb = numb.join("");
    return numb;
},
romanize: function (string){
    var regex = new RegExp('^(M{0,4}(CM|CD|D?C{0,3})|(XC|XL|L?X{0,3})|(IX|IV|V?I{0,3}))$');
    var array = string .split (' ');
    var toreturn = [];
    var half = 0;
    for (var i = 0; i<array.length; i++){
        if (array [ i ].toLowerCase () .includes
            ("second"))
            half = 50;
        if (array [ i ].match (regex)){
            toreturn .push ((this .fromRoman (
                array [ i ]) *100) - 100 + half);
            half = 0;
        }
    }
    131    if (toreturn .length ==0)
        toreturn = [0, 0];
    return toreturn;
},
141

fromRoman: function (str) {
    var tmp = [];
    var toReturn = 0;
    var array = str .split (' ');

```

```

151      for(var i = 0; i<array.length; i++){
152          if(array[i]==='X'||array[i]==='x')
153              tmp[i] = 10;
154          else if(array[i]==='V'||array[i]===
155              'v')
156              tmp[i] = 5;
157          else if(array[i]==='I'||array[i]===
158              'i')
159              tmp[i] = 1;
160      }
161      for(var i = 0; i<tmp.length; i++){
162          if(i!=tmp.length-1){
163              if(tmp[i]>=tmp[i+1]){
164                  toReturn = toReturn + tmp[
165                      i];
166              } else{
167                  var number = tmp[i+1] -
168                      tmp[i];
169                  i = i+1;
170                  toReturn = toReturn+number
171              ;
172          }
173      } else
174          toReturn = toReturn+tmp[i];
175      }
176      return toReturn;
177  },
178
179  date: function(date, on){
180      var todelet = [];
181      if(on) {
182          for (var i = 0; i < date.length; i
183             ++) {
184              todelet.push(false);
185          }
186      }
187      for(var i = 0; i<date.length; i++){
188          if(date[i]==='?'||date[i]==='')
189              date[i] = 0;
190          if(on)

```

```

186           todelet[i] = true;
           } else if(typeof date[i] ===
number' || date[i] instanceof Number){
           } else if(typeof date[i] ===
string' || date[i] instanceof String){
           var matches = date[i].match(/\
d+/g);
           if (matches != null) {
               if (this.about(date[i]))
                   this.data[2].data[i] =
this.data[2].data[i] + "<br><br><br><br><p>style='
color:red'>Data\_incerta.</p>" ;
               if (date[i].includes("-"))
{
               var mol = 1;
               if (this.beforechrist(
date[i]))
                   mol = -1;
               var tmp = date[i].
split("-");
               tmp = this.date(tmp,
false);
               tmp[0] = mol * tmp[0];
               date[i] = tmp;
} else {
               var interval = 0;
               if (date[i].toLowerCase
() . includes("second"))
                   interval = 51;
               } else if (date[i].
toLowerCase() . includes("prima") || date[i].
toLowerCase() . includes("first")){
                   interval = 50;
}
               if (this.beforechrist(
date[i]))
                   date[i] = (-1) *
this.numex(date[i]);
               else

```

```

date[ i ] = this .

numex( date[ i ] ) ;

211      if( interval==51) {
          var num = parseInt
          ( date[ i ] ) ;
          date[ i ] = [ num +
interval , num + 99 ];
          } else if( interval==50)
{
          var num = date[ i ];
          date[ i ] = [ num,
num+interval ];
          }
          }
} else {
          var roman = this .romanize(
date[ i ]) ;
          if( roman===[ 0 , 0 ]&&on)
              todelet[ i ] = true ;
          else if( roman.length ==
2 )
              date[ i ] = [ roman[ 0 ] ,
roman[ 1 ] ];
          else
              date[ i ] = [ roman[ 0 ] ,
this .romaninterval( roman[ 0 ]) ];
          }
} else {
          date[ i ] = 0 ;
          if( on)
              todelet[ i ] = true ;
          }
} }

226      if( on) {
          this .data = this .cleandata( this .
data , todelet );
          }
          return date ;
},
```

```
romaninterval: function (number) {
    if (number%100==50)
        return number+49;
    else
        return number+99;
} ,  
246  
beforechrist: function (date){
    date = date.toLowerCase();
    return (date.includes("a.c.") || date.
includes("ac") || date.includes("bc") || date.includes(
"b.c."));  
} ,  
251  
cleandata: function (data, todelet) {
    for (var i=0; i<todelet.length; i++){
        if (todelet[i]){
            data[1].data[i] = "";
            data[2].data[i]= "";
            data[3].data[i] = null;
        }
    }
    return data;
} ,  
256  
about: function (data){
    var str = data.toLowerCase();
    return (str.includes("circa") || str.
includes("intorno") || str.includes("?") || str.
includes("ca")));
} ,  
261  
Polymer({
    is : 'timeline-datalet' ,
    properties: {
        behavior : {  
271
```

```
276          type : Object,
277          value : {}
278      },
281      data : {
282          type : Array,
283          value : undefined
284      },
286      export_menu : {
287          type : Number,
288          value : 9
289      },
291      timeline : {
292          type : Object,
293          value : {}
294      }
295  },
296  ready: function() {
297      this.behavior = $.extend(true, {}, BaseDataletBehavior, WorkcycleBehavior, AjaxJsonAlasqlBehavior, TimelineBehavior);
298      this.async(function(){this.behavior.
299      init(this)},0);
300      }
301  });
302
</script>
</dom-module>
```

A.2.2 timeline-demo

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title></title>
6     <script>
7       </script>
8   </head>
9   <body>
10  <script src="https://code.jquery.com/jquery-2.1.4.min.
11    js" type="text/javascript"></script>
12  <link rel="import" href="../timeline-datalet.html" />
13  <timeline-datalet data-url="http://hetor.databenc.it/
14    api/3/action/datastore_search?resource_id=871880e3
15    -6278-4684-b728-20fcf6f2b5c0"
16    fields='[]'
17    selectedfields='[{"field":"Column",
18      "value":"Indirizzo", "index":1}, {"field":"Column",
19      "value":"Nome", "index":2}, {"field":"Column",
20      "value":"Descrizione", "index":3}, {"field":"Column",
21      "value":"Coordinate", "index":4}, {"field":"Column",
22      "value":"Anno_di_costruzione", "index":
23      5}]'>
24  </timeline-datalet>
25  </body>
26 </html>
```