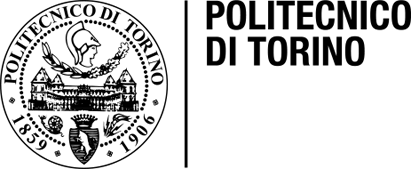
Infrazioni Torino 2017

Visualizzazione dell’Informazione Quantitativa

Immagine che contiene oggetto da esterni

Descrizione generata automaticamente

Valentin Nelu IFRIM S233979

Matteo LATINO S236745

Luca MARONGIU S237973

Indice

[Visualizzazione dell’Informazione Quantitativa 1](https://politecnicoditorinoit-my.sharepoint.com/personal/s233979_politecnicoditorinoit_onmicrosoft_com/Documents/Documenti/Relazione%20VIQ.docx#_Toc11591115)

[1. Introduzione 3](#_Toc11591116)

[2. Analisi Progetto 4](#_Toc11591117)

[2.1. Domande Progetto 4](#_Toc11591118)

[2.2. Lavorazione dati 4](#_Toc11591119)

[2.3. Rappresentazione Mappa 4](#_Toc11591120)

[2.4. Animazioni 5](#_Toc11591121)

[2.5. Rifiniture 5](#_Toc11591122)

[3. Realizzazione 6](#_Toc11591123)

[3.1. Lavorazione dati 6](#_Toc11591124)

[3.2. Rappresentazione Mappa 6](#_Toc11591125)

[3.3. Scelta Grafici 7](#_Toc11591126)

[3.4. Animazioni 7](#_Toc11591127)

[3.5. Rifiniture 8](#_Toc11591128)

[4. Conclusioni 10](#_Toc11591129)

[4.1. Considerazioni e risultato 10](#_Toc11591130)

# Introduzione

Durante il corso di *Visualizzazione dell'Informazione Quantitativa (01QZNxx) c’*è stata la possibilità di svolgere un progetto che ha permesso agli studenti di usare le nozioni apprese nel corso insieme alle conoscenze pregresse di ciascun individuo per poter realizzare un sito web.

Questa breve relazione ha lo scopo di illustrare le scelte fatte e i metodi utilizzati nella realizzazione di tale sito: il suo scopo è quello di mostrare in maniera veloce e intuitiva i dati riguardanti le infrazioni registrate a Torino nel 2017.

# Analisi Progetto

## Domande Progetto

Le domande alle quali si è voluto dare una risposta tramite l’uso della seguente visualizzazione sono:

* Qual è l’andamento nel tempo del numero di infrazioni?
* Quale tipologia di infrazione è la più comune?
* Qual è la distribuzione delle infrazioni tra i vari quartieri?

L’ analisi relativa ai primi due quesiti è stata inoltre eseguita sia in relazione alla città di Torino nella sua interezza, sia in relazione ai singoli quartieri.

## Lavorazione dati

Lo scopo principale del progetto è quello di rappresentare con dei grafici i dati presenti su due file CSV:

* <http://www.comune.torino.it/opendata/VRC%2001072017%2031122017.csv>
* <http://www.comune.torino.it/opendata/VRC2017_gen_giu.csv>

I primi tre campi sono necessari alla rappresentazione della data del verbale, che viene utilizzata anche come chiave di ordinamento. La tipologia di verbale è al 97% “POLIZIA MUNICIPALE” perciò questo parametro è stato escluso dal grafico. In seguito, c’è il tipo di sanzione che può assumere due soli valori (Amministrativa / Tributaria).

I campi a seguire indicano la via e il numero civico di dove è stata emessa la sanzione. Il campo “Via 2” è stato rimosso poiché interamente vuoto e superfluo per il progetto stesso. I dettagli riguardo la descrizione della sanzione sono stati anch’essi rimossi poiché troppo descrittivi e impossibili da schematizzare, inoltre non rispondono alle domande poste. L’ultimo campo presente nei due CSV è il numero di verbali che verrà in seguito utilizzato come parametro moltiplicativo.

Per poter ottenere una divisione in quartieri è stato necessario l’utilizzo di un terzo file [CSV](http://aperto.comune.torino.it/dataset/toponomastica) che crea una corrispondenza tra il singolo indirizzo e il quartiere, utilizzando l’informazione del CAP.

## Rappresentazione Mappa

La mappa di Torino svolge principalmente due compiti:

* Permettere all’utente di selezionare i quartieri, fino ad un massimo di tre, dei quali se ne vuole conoscere le informazioni in maniera più dettagliata, o dei quali se ne vuole eseguire un confronto. I grafici che vengono mostrati in seguito alla selezione sono: un andamento nel tempo delle infrazioni e la percentuale delle tipologie di infrazioni dei quartieri selezionati (Tributarie o Amministrative).
* Rappresentare in maniera qualitativa la distribuzione del numero di infrazioni totali nei vari quartieri della città, attraverso l’uso di una variazione della tonalità del colore del quartiere.

## Animazioni

Per quanto riguarda le animazioni ciò che si è fatto è stato concentrarsi sull’*hover* sulla mappa di Torino e sui click nei singoli quartieri.

È stato fondamentale far distinzione all’interazione con i singoli poligoni e con i relativi testi rappresentanti il nome del quartiere essendo elementi distinti nel file *html*, ma essenzialmente le operazioni eseguite sono state le stesse per entrambi.

Quando il cursore passa sopra il singolo poligono o sulla relativa scritta il poligono rappresentante il quartiere cambia colore, passando dal suo colore originario ad un *‘lightblue’*, mentre il bordo che lo circoscrive si inspessisce passando ad una grandezza di 10 pixel (rispetto ai 2 pixel che lo costituiscono di default), ciò che accade alle scritte invece è che esse vengono ingrandite a 22pt a discapito dei 14pt di default.

Quando il cursore esce dal poligono invece le caratteristiche originarie del grafico vengono ripristinate.

Se viene effettuato un click sul quartiere (o sul suo nome) l’animazione che permette all’utente di riconoscere ed individuare i quartieri scelti è incentrata sui loro nomi, ciò che avviene è un inspessimento della scritta e un cambio di colore che passa dal nero di default ad un con bordo nero per ogni singola lettera, la selezione dei quartieri è limitata a 3 quartieri, questo per rendere comprensibili i grafici e agevolare il confronto, se questo numero viene oltrepassato l’utente visualizzerà un messaggio da parte del browser che lo informa di tale caratteristica del sito.

Quando viene effettuato un secondo click sullo stesso quartiere esso viene deselezionato per quanto riguarda la visualizzazione dei relativi grafici e dunque vengono ripristinate le caratteristiche originarie del quartiere sia per il colore che per la grandezza delle scritte.

Tutte le scelte effettuate per le animazioni, sia per i colori che per gli inspessimenti di testo e bordi, sono state pensate per rendere all’utente una facile visualizzazione del quartiere in questione e quindi una conseguente facilità all’utilizzo del sito stesso.

Per rendere il sito mutevole e adattabile a tutte le dimensioni della finestra del browser si è pensato anche ad una variazione della grandezza di tutti i grafici presenti nella pagina.

Ciò che si è pensato è appunto un adattamento della grandezza dei grafici a quella della finestra del browser, infatti quando la larghezza di questa è inferiore a 800 pixel (larghezza delle visualizzazioni di default) ogni singolo grafico viene visualizzato per l’intera larghezza della pagina, mentre se la larghezza è superiore a 800 pixel la mappa di Torino viene visualizzata affianco ai grafici del tempo e della tipologia di infrazioni occupando il 45% dell’intera larghezza (lo stesso discorso si applica alle altezze), l’unico grafico che mantiene la larghezza dell’intera pagina è quello che rappresenta il numero di infrazioni per tutti i quartieri della città.

## Rifiniture

Soddisfatte le richieste principali è stato necessario rivedere la scelta dei colori, le dimensioni dei grafici e garantire il corretto funzionamento dell’intero sito: quindi sistemare gli ultimi bug ed eseguire il *refactoring* del codice.

# Realizzazione

## Lavorazione dati

Come accennato nell’analisi dei dati molti campi sono risultati inutili ai fini del progetto perciò è stato necessario ripulirli. Il primo passo è stato quello di unire i due CSV contenenti le infrazioni in un unico file tramite un banale *copia-incolla*. Sono stati tentati più approcci uno dei quali consisteva nel convertire inizialmente il tutto in un file JSON e poi implementare tutta la logica del programma in *JavaScript,* questo approccio è stato subito scartato poiché il file diventava pesante e lento da processare.

L’approccio definitivo è stato quello di snellire i CSV tramite degli script *Python 3*, *01a\_Snellisci\_VRC.py* e *01b\_Snellisci\_NCC.py,* con lo scopo di rimuovere i campi non utilizzati e creare delle versioni CSV *\_min.* Il file contenente i CAP è stato ulteriormente ridotto cercando di aggregare le parte delle vie in base al numero civico: lo script *02\_Crea\_min\_max\_civico.py* prende in ingresso il CSV ridotto e dopo aver ordinato le singole istanze in un vettore le raggruppa ove possibile. In fine tramite l’ausilio di *03\_Converter.py* i due file CSV sono stati convertiti in due rispettivi file JSON. Per completare la preparazione dei dati è stato utilizzato *04\_Data\_merger.py*: questo script, dopo aver caricato in memoria i due file ottenuti dal passaggio precedente associa ad ogni oggetto sanzione un CAP. Quest’ultimo processo ha richiesto delle rifiniture poiché non tutte le vie presenti nel file delle infrazioni trovavano una corrispondenza adeguata (ad esempio nel primo file c’era “Corso A. Tassoni” mentre nel secondo “Corso Alessandro Tassoni”): le modifiche apportate a mano sono state fatte tramite l’ausilio di Visual Studio Code che ha permesso di rinominare più istanze contemporaneamente. Le modifiche sono state fatte sui file CSV quindi prima di unire i dati è stato necessario eseguire nuovamente la conversione in JSON.

Alle sanzioni che non presentavano alcuna via è stato associato un campo *vuoto* come CAP: queste infrazioni sono state ignorate nella rappresentazione vista l’impossibilità di attribuirle ad uno specifico quartiere e visto il loro numero molto esiguo.

## Rappresentazione Mappa

La mappa rappresentante la città di Torino è stata prodotta tramite l’uso dei tag html *<svg>, <polygon>* e *<text>.* Tramite il tag *<svg>* è infatti possibile dichiarare un’area all’interno della quale poter graficare la mappa le cui dimensione sono in grado di adattarsi alla dimensione della finestra, mentre tramite il tag *<polygon>* sono stati modellati i vari poligoni rappresentanti ciascuno un singolo quartiere della città. La modellizzazione è stata eseguita ricalcando i confini dei vari quartieri da un file immagine contenente la suddivisione in quartieri della città, per eseguire tale operazione si è fatto uso del sito [https://www.image-map.net](https://www.image-map.net/) , attraverso il quale è stato possibile ottenere le informazioni necessarie per la definizione della forma dei vari poligoni. Tramite il tag *<text>* sono stati inseriti sulla mappa i nomi dei vari quartieri.

La colorazione dei vari quartieri in base al numero di infrazioni viene eseguita dalla funzione *mapColor().* Questa funzione dopo aver estratto dal vettore “quartieri” le informazioni relative al numero di infrazioni le mappa su una scala comprendente il range di valori [0 - 255], il valore così ottenuto viene utilizzato per regolare la saturazione del verde, di conseguenza tenendo costante a 255 la saturazione del rosso e a 0 la saturazione del blu è possibile di rappresentare il N° di infrazioni sullo spettro di colori fra il giallo ed il rosso.

Infine, accanto alla mappa, in basso a destra, attraverso l’uso del tag *<* *rect >* è stata inserita una legenda indicante il metodo con cui interpretare i vari colori, ossia colori tendenti al giallo equivalgono a poche infrazioni ≈ 50 - 100, mentre per colori tendenti al rosso equivalgono un numero di infrazioni maggiori ≈ 900 - 1000.

## Scelta Grafici

La realizzazione dei grafici è stata eseguita tramite l’uso della libreria *PLOTLY* di Javascript, ogni grafico punta a rispondere ad una specifica domanda.

Per visualizzare “l’andamento nel tempo del numero di infrazioni” abbiamo utilizzato un diagramma a linee con il tempo, suddiviso in mesi, sull’asse X e il numero di infrazioni sull’ asse Y. Su tale grafico è possibile visualizzare o l’andamento relativo all’intera città di Torino o quello dei singoli quartieri, la selezione di una o dell’altra informazione viene eseguita dall’utente attraverso l’uso della mappa di Torino.

Nel caso della “tipologia di infrazione è la più comune” si è invece optati per un diagramma a barre sovrapposte-a percentuale, si è scelto tale tipo di grafico in quanto dato che le tipologie di infrazioni possibili risultano essere solo due: “Amministrative” o “Tributarie”; risulta possibile eseguire in maniera abbastanza semplice sia confronti fra la percentuale di una o dell’altra tipologia in un singolo quartiere, sia il confronto fra come varia tale suddivisione fra i vari quartieri. Anche per questo grafico è possibile visualizzare le informazioni relative all’intera città di Torino o quello dei singoli quartieri.

In entrambi i casi è possibile confrontare fra loro le informazioni relative massimo a tre quartieri: si è scelto tale vincolo per evitare un numero eccessivo di informazioni su un singolo grafico, il quale altrimenti diventerebbe di difficile lettura; inoltre tali grafici variano dinamicamente in base ai quartieri selezionati dall’utente.

Il “Numero di infrazioni totali in un quartiere” è stato invece rappresentato tramite un diagramma a barre verticali, ed ordinato in maniera decrescente in base al numero di infrazioni del singolo quartiere. Su tale grafico è inoltre indicata la media del numero di infrazioni.

## Animazioni

Tutte le animazioni riguardanti la mappa di Torino sono incluse nel file ‘*animazioni.js’*.

La funzione che regola l’interazione tra l’utente e il singolo quartiere è ‘*set\_page\_event\_listeners()’* che in cui all’inizio vengono salvati i set dei poligoni riguardanti i quartieri e i relativi nomi in due differenti vettori, dopo di che viene effettuato un ciclo sui entrambi definendo degli event listeners che gestiscono l’interazione con l’utente:

* Il primo event listener è quello per il *‘mouseover’* in cui sia per i poligoni che per i testi l’operazione eseguita è stata quella di cambiare il colore del poligono, la larghezza dei bordi e la grandezza dei font dei nomi dei quartieri, tutto ciò è stato fatto tramite *JQuery* e in particolare tramite il metodo css , la sintassi è stata : *$(polys[i]).css(…).*Il problema che abbiamo incontrato in questa interazione è stata che i bordi di confine tra due quartieri tra cui quello in hover non rispettassero la larghezza imposta dall’evento, questo a causa della sequenzialità dei poligoni nel codice html: capitava quindi che i bordi dei quartieri confinanti andassero a sovrapporsi non facendo mostrare così l’inspessimento aggiunto dall’evento, perciò l’ulteriore operazione eseguita è stata appendere come figlio all’intera mappa il quartiere in questione e il relativo nome in modo che tutti i bordi rispettassero la modifica eseguita, la sintassi è stata : ‘*svg.appendChild(…)’* in cui svg è la mappa di Torino.
* L’evento che gestisce la selezione dei quartieri è quello per il *‘click’*, in cui viene modificata la grandezza dei font delle scritte e il colore delle scritte aggiungendo inoltre una *‘text-shadow’* nera di 1 pixel alle scritte in modo che esse vengano risaltate nonostante il colore di sfondo del quartiere. Questo evento deve tener conto dei quartieri già selezionati in modo che quando viene effettuato un click su uno di essi vengano ripristinate le caratteristiche originarie, ciò è stato eseguito tramite un vettore denominato ‘*selected’* che tiene conto appunto di questi quartieri, la funzione che controlla se un quartiere è stato già selezionato o meno è ‘*isIn(…)’* che riceve come unico parametro l’id del quartiere di cui si vuole verificare la già avvenuta selezione.
* L’ultimo evento che gestisce le animazioni per quanto riguarda l’interazione tra mappa e utente è quello che regola il *‘mouseleave’* che ha l’unico compito di ripristinare le caratteristiche originali e togliere le modifiche fatte dall’hover.

Tutti gli eventi riguardanti le animazioni dei quartieri tengono conto di quelli già selezionati tramite la funzione già introdotta *‘isIn(…)’*.

La funzione che invece regola la dimensione dei grafici in relazione a quella della finestra del browser è *‘setta\_responsivita()’* ogni qual volta il browser cambia dimensione (*window.onresize(…)’*), ciò che fa è impostare la larghezza e l’altezza dei grafici in base a quella della finestra impostando diversi valori delle dimensioni dei grafici in base a quelle della finestra del browser (se è più grande o più piccolo di 800 pixel).

Quest’ultimo evento è pensato in modo che l’utente riesca in modo agevole a visualizzare i grafici anche da uno smartphone o da un dispositivo con un display non eccessivamente grande.

## Rifiniture

Conclusi tutti gli step si è arrivati ad un modello funzionante ma con alcuni bug e ritardi. Uno dei principali problemi da risolvere era il mancato caricamento dei dati all’primo avvio del sito. A causa della lettura asincrona dei dati e del tempo impiegato dal browser ad eseguire gli script necessari a popolare il vettore dei quartieri con le rispettive infrazioni, i grafici e la colorazione della mappa non venivano eseguiti. Per risolvere questo problema è stato deciso di utilizzare un approccio “sequenziale”: al caricamento della pagina viene chiamata la funzione *readQuartieri()* chein primis avvia il meccanismo di lettura del file JSON; una volta conclusa la lettura dei file vengono immediatamente eseguita la funzione *mapColor()*, la quale contiene al suo interno le chiamate a *colorQuartieri()* e *setta\_responsivita()* che si occupano della colorazione della mappa e del disegno dei grafici in base alle dimensioni della pagina.

Per permettere una discreta visualizzazione anche su dispositivi mobile, oltre alla funzione *setta\_responsivita()* sopra citata, sono state utilizzate due *media query* nel css: la prima entra in gioco per dispositivi con uno schermo largo almeno 800px (in generale desktop); la seconda invece serve per i dispositivi mobile orientati verticalmente. Per poter garantire il giusto posizionamento della mappa di Torino e dei due grafici utilizzati per il confronto è stata utilizzata la proprietà *display:flex* che ha permesso un agevole manipolazione degli elementi. Quest’ultima scelta è stata imposta dai fallimentari e macchinosi utilizzi della proprietà *float* del css.

# Conclusioni

## Considerazioni e risultato

Il risultato di questo progetto è un sito web che permette all’utente di effettuare confronti tra le informazioni fornite dalle mappe e quindi trarre considerazioni riguardanti caratteristiche della città di Torino (anche se i dati utilizzati sono relativi esclusivamente al 2017), l’obiettivo del progetto è stato raggiunto in quanto si è riuscita a creare una mappa di Torino dinamica e tramite essa è possibile visualizzare grafici.

Il sito risulta facile da utilizzare e molto intuitivo, comunque sia è presente una piccola descrizione sul suo utilizzo ad inizio della pagina stessa.

Questo progetto ci ha permesso di apprendere abbastanza a fondo il linguaggio di programmazione Javascript e di impararne un utilizzo nella creazione, modifica e analisi di grafici di varie tipologie; abbiamo anche sviluppato e ampliato le nostre capacità di teamworking e cooperazione riuscendo a suddividerci i compiti in maniera adeguata e coordinata.

Per lo sviluppo del progetto abbiamo utilizzato i software *Webstorm* e *Visual Studio Code* perla scrittura del codice, per il versioning abbiamo usato un repository *GitHub* che ci ha permesso un agevole coordinamento. Ci siamo tenuti in contatto tramite un gruppo Telegram e periodiche riunioni per discutere le nostre idee e l’evoluzione del progetto stesso.

Abbiamo partecipato attivamente tutti e tre alla scrittura del codice risolvendo a vicenda dubbi e bug riscontrati nei codici.

In particolare, le responsabilità di ognuno di noi sono state:

* Valentin Nelu IFRIM:
  + Lavorazione dati
  + Creazione struttura di base
  + Script per animazioni e responsiveness
  + CSS
  + Bug Fixing
* Matteo LATINO:
  + Creazione codice HTML per la mappa di Torino
  + Creazione e gestione dei singoli Grafici
  + CSS
  + Bug Fixing
* Luca MARONGIU:
  + Reperimento file necessari, in particolare <http://aperto.comune.torino.it/dataset/toponomastica> (terzo file CSV sopra menzionato)
  + Creazione e gestione classe Quartieri
  + Script per animazioni e responsiveness
  + CSS
  + Bug Fixing

Come sopra riportato è da notare che la suddivisione dei compiti non è stata così rigida, ognuno ha contribuito equamente allo sviluppo e al miglioramento delle diverse parti del sito, nell’idea e nella realizzazione.