Alma Mater Studiorum
Università di Bologna
Campus di Cesena

SCUOLA DI SCIENZE
Corso di Laurea in Ingegneria e Scienze Informatiche

Sviluppo di un'applicazione web per la visualizzazione interattiva di dati eterogenei

Relazione finale in

Tecnologie Web

Relatore:
Dott.ssa Catia Prandi

Presentata da:
Andrea Betti

Correlatore:
Dott.ssa Chiara Ceccarini

Sessione II
Anno Accademico 2018-2019

Introduzione

Nel corso del 2019, il Consiglio degli Studenti e le associazioni studentesche dell'Università di Bologna hanno condotto un'indagine denominata HousINg Unibo sulla condizione abitativa degli studenti iscritti presso l'Alma Mater Studiorum - Università di Bologna, al fine di individuare soluzioni per migliorare la condizione abitativa degli studenti.

I dati ricavati dalle compilazioni del questionario, raccolti in forma anonima e seguendo i principi del regolamento generale sulla protezione dei dati, danno la possibilità di evidenziare quanti studenti si sono trasferiti a Bologna per frequentare le lezioni tramite il confronto delle località di domicilio con quelle di residenza e con il luogo di nascita. Inoltre, è possibile ricavare informazioni sui percorsi di studi da loro intrapresi, sulla tipologia di abitazione in cui vivono (in affitto, in studentato...) e su quanti di essi pensano di trasferirsi o continuare a vivere a Bologna.

Questo progetto nasce per cercare di comunicare in modo efficace il significato di questi dati attraverso rappresentazioni visuali e interattive.

È stata sviluppata un'applicazione web che fa uso della libreria React per la creazione delle interfacce relative a una mappa geografica e a una serie di grafici a barre e a torta, che racchiudono rappresentazioni grafiche dei dati degli studenti memorizzati in un database MongoDB, interrogato dall'applicazione attraverso l'uso della libreria Mongoose. Viene utilizzato un server creato utilizzando il framework Koa.js, che tramite route apposite

può restituire i dati archiviati e inizializzare i dati del database basandosi sulle informazioni contenute in file JSON, a loro volta convertiti da file CSV mediante appositi script Python.

Tramite un processo di geocodifica, sono state ricavate le coordinate delle località associate agli studenti a disposizione per rendere possibile la rappresentazione grafica dei dati sulla mappa mediante diverse modalità: marcatori, mappa di calore, mappa coropletica dei quartieri di Bologna. L'utilizzo di mappe per la rappresentazione di dati è molto efficace perché possono essere interpretate facilmente.

È stata data all'utente la possibilità di selezionare la modalità di visualizzazione e il tipo di posizione dei dati degli studenti da visualizzare, sia per quanto riguarda la sezione dell'interfaccia relativa alla mappa che per quella relativa ai grafici, permettendogli di visualizzare solo la porzione di dati che gli interessa e senza fruire passivamente di tutte le informazioni in una pagina statica.

I dati possono essere filtrati, oltre che per diverse proprietà proprie degli studenti come l'anno di corso e l'anno di nascita, anche per la distanza dalla sede del corso frequentato dallo studente. Le coordinate delle sedi dei corsi a cui viene fatto riferimento per misurare le distanze sono state ricavate consultando gli indirizzi riportati nelle pagine dei diversi corsi dell'Università di Bologna, e successivamente memorizzati anch'essi nel database MongoDB come per le informazioni degli studenti.

La tesi è strutturata in tre capitoli:

- Il primo capitolo descrive i concetti delle principali tematiche che hanno interessato lo sviluppo dell'applicazione: dati spaziali, visualization, big data e open data.

- Il secondo capitolo analizza i linguaggi e le principali tecnologie utilizzate all'interno del progetto.

- Il terzo capitolo descrive le tecniche di implementazione impiegate nelle diverse parti dell'applicazione.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Contesto del progetto

In questo capitolo vengono riportati i concetti principali riguardanti tematiche che hanno interessato lo sviluppo di questo progetto di tesi, che riguarda l'analisi e la visualizzazione interattiva di dati eterogenei nello specifico contesto dell'Università di Bologna.

In particolare, si inizierà descrivendo cosa sono i dati spaziali e alcuni dei processi che ne fanno uso. Verrà poi introdotto il concetto di visualization e delle sue principali suddivisioni, tra cui ha rilievo la data visualization. Infine, si parlerà di big data, open data e di come è opportuno trattarli nello sviluppo di applicazioni.

1.1 Dati spaziali

Per **dati spaziali**, ci si riferisce a dati che contengono informazioni relative a coordinate geografiche. Possono assumere diversi formati e contenere più informazioni su posizioni specifiche.

L'utilizzo più comune di dati spaziali è attraverso un **GIS** (**Geographic Information System**, in italiano **sistema informativo geografico**), ovvero un programma o un insieme di programmi che lavorano insieme per dare un significato a questi dati mediante operazioni di gestione, manipolazione, personalizzazione, analisi e creazione di rappresentazioni visuali. L'utente

medio usa più collezioni di dati (dataset) spaziali contemporaneamente e li confronta o unisce tra loro; ciascun dataset spaziale può essere definito layer (strato).

Esempi di layer diversi sono i dati relativi a strade, quartieri e posizioni di luoghi specifici, rappresentabili rispettivamente da linee, poligoni e punti. Il posizionamento dei layer è importante ai fini della visualizzazione, in quanto aiuta la comprensione dei diversi tipi di dati e rende più chiara la comunicazione delle informazioni. Ad esempio, è ragionevole dare la priorità al layer delle posizioni su quello delle strade, in quanto altrimenti queste ultime potrebbero non essere visibili.[1]

1.1.1 Geocodifica

È possibile ricavare dati spaziali attraverso il processo di **geocodifica**: esso trasforma la descrizione di un luogo, come un indirizzo o un nome, in coordinate geografiche che possono essere utilizzate per operazioni di mappatura o analisi spaziale.

Le descrizioni geografiche possono essere ricavate da molte fonti, sia private che pubbliche, e sono tipicamente conservate in formati tabellari. I dati da geocodificare devono essere preparati e trasformati in una semplice tabella di dati che possa essere inviata con facilità a un localizzatore di punti o indirizzi. È possibile, ad esempio, che fogli di calcolo contenenti degli indirizzi abbiano al loro interno anche dati estranei o metadati da rimuovere o organizzare per creare una semplice tabella che possa essere importata in un formato di dati geospaziali, come Shapefile o File Geodatabase.

I dati da geocodificare vengono poi inoltrati a un locator, un software specializzato che utilizza dati georeferenziati, come strade o posizioni di luoghi conosciuti, per creare dei legami tra le descrizioni di input e delle coordinate geografiche. I dati georeferenziati sono progettati e organizzati per un tipo di locator specifico. La qualità del locator è dipendente dall'attualità e dalla precisione dei dati referenziati di cui fa uso.[2]

Dopo il completamento del processo di geocodifica, è raccomandabile controllare il grado di precisione del risultati. I risultati geocodificati relativi a punti specifici spesso non soddisfano le aspettative: la probabilità che l'indirizzo di input non riesca a essere geocodificato varia dal 10% al 30%, e quando viene restituito un risultato sono comuni errori posizionali di diverse centinaia di metri. L'incompletezza e l'inesattezza di dati geocodificati ha un effetto avverso su analisi statistiche e può portare, se ignorato, a conclusioni di valore dubbio. Devono essere esercitate metodologie statistiche di analisi di dati mancanti o imprecisi per far fronte a problemi di geocodifica incompleta e incorretta, e permettere di trarre conclusioni statisticamente valide dai dati.[3]

1.1.2 Georeferenziazione

La **georeferenziazione**, a differenza della geocodifica, consente di associare a un dato delle coordinate che ne fissano la posizione geografica e altri attributi relativi, come ad esempio la categoria di un edificio, il suo anno di costruzione, ecc.

La georeferenziazione viene applicata tipicamente a dati di tipo vettoriale e di tipo raster. I primi consistono in rappresentazioni grafiche semplificate del mondo reale, e possono consistere in punti (georeferenziazione puntuale), linee (georeferenziazione lineare) o poligoni (georeferenziazione areale)[4]. I vettori sono considerati il modo più efficace per rappresentare in modo semplicistico gli elementi sulla superficie terrestre. I raster sono invece presentati in una griglia di pixel. Ciascun pixel ha un valore, che sia un colore o un'unità di misura, che comunica informazioni sull'elemento in questione. Nel contesto dei dati spaziali, consistono in foto del mondo reale scattate da satelliti o altri dispositivi aerei. La qualità dei dati raster dipende dalla loro risoluzione.[1]

Vi sono due metodi fondamentali di georeferenziazione[5]:

Figura 1.1: Confronto tra un dato di tipo vettoriale e un dato di tipo raster rappresentanti la stessa superficie.[1]

- **sistema geocentrico**: utilizza un sistema di coordinate tridimensionale con il centro della Terra come origine dei tre assi. Questo metodo è utilizzato universalmente in applicazioni scientifiche, ma non è facile da usare quando applicato a punti della superficie terrestre perché gli assi del sistema sono paralleli agli assi di rotazione della Terra piuttosto che ai punti cardinali. Il sistema può essere espresso sia come un sistema di coordinate cartesiano tridimensionale in forma xyz che con longitudine (λ), latitudine (φ) e altezza (H) sopra una superficie di riferimento conosciuta.

- **proiezione**: le tre coordinate vengono espresse come un piano, con l'altezza sopra di esso. In altre parole, la superficie terrestre viene rappresentata in un piano. Le coordinate individuano l'esatta posizione di un punto compreso nella proiezione e sono misurate come distanze, est e nord dall'origine.

Il metodo più comune per esprimere l'altezza è attraverso il livello del mare. Tuttavia, il suo calcolo risulta complesso perché si trattano valori

irregolari e in costante cambiamento. Per far fronte a questi inconvenienti, gli scienziati hanno inventato una superficie regolare chiamata elissoide per misurare il livello del mare. A causa della complessità della forma del livello del mare, sono richieste diverse centinaia di ellissoidi in proporzione alla zona terrestre che si vuole rappresentare.

1.2 Visualization

La **visualization**, o **visualizzazione**, è l'insieme di tecniche utilizzate per rappresentare un oggetto, una situazione o un concetto astratto attraverso immagini. Il suo obiettivo principale è quello di analizzare, esplorare, scoprire, illustrare e comunicare le informazioni in una forma ben comprensibile. Queste devono essere presentate coerentemente, compattamente e da punti di vista diversi, fornendo molteplici livelli di dettaglio.[6]

L'approccio nella progettazione di visualizzazioni organizzate può essere suddiviso in sei passi[7]:

1. **mappatura**: si intende il modo in cui le informazioni vengono convertite in un risultato grafico. Deve essere definita e seguita in modo consistente in tutta l'applicazione, cercando di dare un rilievo agli aspetti concettualmente importanti.

2. **selezione**: consiste nel definire quali dati sono rilevanti alla visualizzazione. Da un lato, visualizzare dati insufficienti può portare gli utenti a prendere decisioni non ottimali o completamente sbagliate; dall'altro, visualizzare troppi dati non necessari può complicare la comprensione della rappresentazione.

3. **presentazione**: è necessario prendere in considerazione anche le dimensioni del dispositivo su cui può essere osservata la visualizzazione per gestire e organizzare le informazioni nel modo più efficiente possibile.

4. **interattività**: la possibilità di interagire con gli elementi grafici migliora il coinvolgimento dell'utente nell'osservazione e nella fruizione dei dati visualizzati.

5. **fattori umani**: una visualizzazione deve essere osservabile velocemente dall'occhio umano e facilmente interpretabile dagli utenti, in modo che la conoscenza sulla percezione visiva umana e sugli aspettivi cognitivi possa facilitare la progettazione di una visualizzazione efficace.

6. **valutazione**: dopo avere ultimato la creazione di interfacce di visualizzazione, è importante effettuare dei test per verificare il livello di efficacia del risultato ottenuto.

Vi sono due principali categorie di visualizzazione: l'**infografica** e **data visualization**. Le infografiche sono rappresentazioni visuali soggettive di fatti, eventi o numeri che riflettono dei modelli e hanno lo scopo di raccontare una storia e portare l'osservatore alle conclusioni dell'autore; sono create manualmente per uno specifico insieme di dati. Per data visualization, o visualizzazioni di dati, si intende la rappresentazione grafica di dati al fine di rendere più comprensibile e veloce la fruizione delle informazioni relative. Sono ottimali per guidare l'osservatore a trarre le proprie conclusioni dai dati osservati, e sono generati automaticamente per dataset arbitrari.

Un'infografica può contenere visualizzazioni di dati, ma non viceversa.[8]

La tabella seguente riassume in linee generali le principali differenze tra le due tipologie di visualization.[9]

1.2.1 Data visualization

I dati consistono in nomi, quantità, posizioni e altri tipologie di contenuti testuali e numerici tipicamente conservati sotto forma di tabelle, con righe di record e colonne di diverse variabili.

Ciò che il creatore di una visualizzazione di dati vuole comunicare con essi condiziona la modalità di rappresentazione: attraverso le tabelle, è possibile

	Data visualization	Infografica
Metodo di generazione	Uso maggiore di numeri	Creazione di immagini originali
Quantità di dati	Più dati	Meno dati, più conclusioni
Grado di cura estetica	Meno originale, più focalizzata sulle informazioni	Più originale
Grado di interattività	Interattiva (variazione di dati)	Statica (dati fissi)

Tabella 1.1: Confronto tra data visualization ed infografica

avere una lettura visiva e immediata dei singoli dati individualmente, ma non sono adatte per mostrare confronti tra molteplici valori. La mente umana non è in grado di dare facilmente a del contenuto testuale un significato quantitativo e qualitativo, ed è per questo motivo che per raggiungere tale scopo vengono utilizzati oggetti visuali come barre, linee, punti, ecc.

Osservando una data visualization, l'individuo attraversa tre fasi:

- **percezione**: attraverso la prima lettura del grafico, analizza la forma, la dimensione e il colore delle rappresentazioni dei dati e converte queste informazioni in valori percepiti.
- **interpretazione**: dopo avere assimilato informazioni dalla percezione del grafico, cerca di assegnare loro un significato; l'interpretazione è condizionata dalla conoscenza preliminare dell'oggetto rappresentato e dalla capacità di associarla a ciò che viene letto. Il creatore del grafico ha il compito di guidare l'osservatore a interpretare la rappresentazione attraverso scelte di design adatte.
- **comprensione**: dopo avere percepito e interpretato la data visualization, l'osservatore si interroga se avere appreso queste informazioni

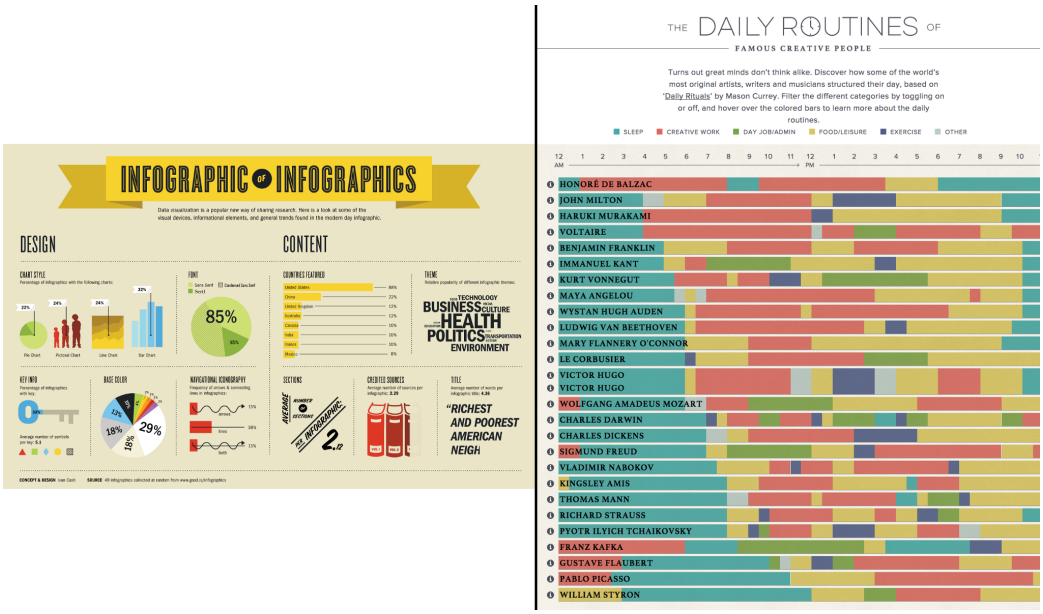

Figura 1.2: Un esempio di un'infografica e un esempio di una visualizzazione di dati a confronto.[8]

abbia avuto una rilevanza personale.[10]

Il campo di data visualization prende spunti da molte discipline come, ad esempio, dalla psicologia, per lo studio della percezione dei dati e dell'impatto di alcuni elementi (come forme e colori), dalle scienze informatiche e dalla statistica, che hanno sviluppato diversi nuovi settori come machine learning e tecniche di data mining, e dalla grafica e dal design multimediale per la costruzione di risultati visuali.[11]

Le aziende, attraverso rappresentazioni grafiche delle informazioni sulla loro attività, sono in grado di vedere grandi porzioni di dati in modo chiaro e coeso, traendone delle conclusioni più velocemente. Anche grandi porzioni di dati complicati possono essere compresi più facilmente se presentati graficamente, con la possibilità di riconoscere parametri altamente correlati tra loro, la cui relazione non sarebbe stata notata altrimenti. Identificare queste relazioni aiuta le organizzazioni a concentrarsi su settori che possono portare

facilmente a raggiungere gli obiettivi più importanti.

Utilizzando strumenti di data visualization per scoprire nuove mode può dare alle aziende la possibilità di avere vantaggi sulla concorrenza. È facile notare anomalie che condizionano la qualità del prodotto o la perdita di clienti, e quindi avere la possibilita di agire in tempo per evitare che i problemi si ingrandiscano.

Una volta che un'impresa ha scoperto nuove intuizioni da delle analisi visuali, il passo successivo è comunicarle ad altri. L'utilizzo di strumenti di data visualization è importante perché incuriosisce e comunica il messaggio velocemente.[12]

Per evitare insidie nel settore di data visualization è opportuno scegliere gli strumenti giusti per raggiungere l'obiettivo desiderato. È essenziale che l'azienda abbia in mente una buona idea di chi possano essere gli utenti potenziali e di come questi utilizzeranno ciò che è stato dato loro a disposizione.[13]

1.2.2 Tipologie di diagrammi

Vi sono diverse tecniche di data visualization convenzionali con caratteristiche generiche e di comprensione comune, dipendenti dalla tipologia di dati e dall'obiettivo della rappresentazione[6]:

- **tabella**: è una tecnica di rappresentazione di dati facili da capire ed interpretare. È un formato strutturato, organizzato da righe e colonne che formano relazioni. Solitamente le righe rappresentano variabili indipendenti nei cosiddetti record o tuple, mentre le colonne sono variabili dipendenti.

- **grafico a torta**: un cerchio che descrive una quantità specifica divisa in settori, o fette, ciascuno dei quali rappresenta dati con caratteristiche comuni. Per rendere più chiara la quantità che ciascuna fetta rappresenta, è possibile associare delle etichette con la precisa quantità o la percentuale rispetto al totale. È importante assegnare un colore diverso

per ciascun segmento in modo da rendere più facilire la comprensione e l'interpretazione.

- **grafico a barre**: è usato principalmente per rappresentare dati discreti, che possono assumere un numero finito di valori, piuttosto che dati continui, compresi in un range definito e potenzialmente infiniti. L'asse della dimensione mostra le voci della categoria che vengono confrontate, mentre l'asse della misura mostra il valore per ciascuna voce della categoria. Il grafico a barre può essere visualizzato in senso orizzontale o verticale.[14] Per ciascuna categoria possono essere associate più barre relative a diverse serie di dati, affiancandole o sovrapponendole tra di loro dando la priorità di visualizzazione a quelle associate ai valori più bassi.

- **istogramma**: introdotto dal matematico Karl Pearson, è un tipo di grafico utilizzato per rappresentare la distribuzione di dati numerici su un intervallo continuo. A ciascun intervallo corrisponde una barra, la cui altezza è correlata alla relativa frequenza assoluta del dato; solitamente vengono utilizzati intervalli uniformi, ma è possibile rappresentare intervalli di diversa lunghezza.

- **grafico a linee**: è utilizzato per visualizzare l'andamento dei dati nel corso del tempo, utilizzando punti interconnessi tra linee continue o rette. È possibile evidenziare la categoria del dato assegnando ai punti delle icone.

- **grafico ad aree**: è un altro tipo di grafico utilizzato per mostrare la grandezza della variazione nel tempo di dati quantitativi. Viene utilizzato come base il grafico a linee, e vengono riempite le aree sottostanti a ciascuna linea assegnando loro un colore diverso.

- **grafico a dispersione**: viene mostrato in un piano cartesiano il rapporto tra due variabili, una dipendente e una indipendente, di ciascun

punto dato. Mostra quanto è forte la relazione tra le variabili e lo stato di dispersione dei dati.

- **grafico a bolle**: è una variazione del grafico a dispersione che può essere utilizzato con dei punti dati composti da tre variabili. La terza dimensione è rappresentato dall'ampiezza della circonferenza il cui centro è determinato dalle altre due dimensioni.

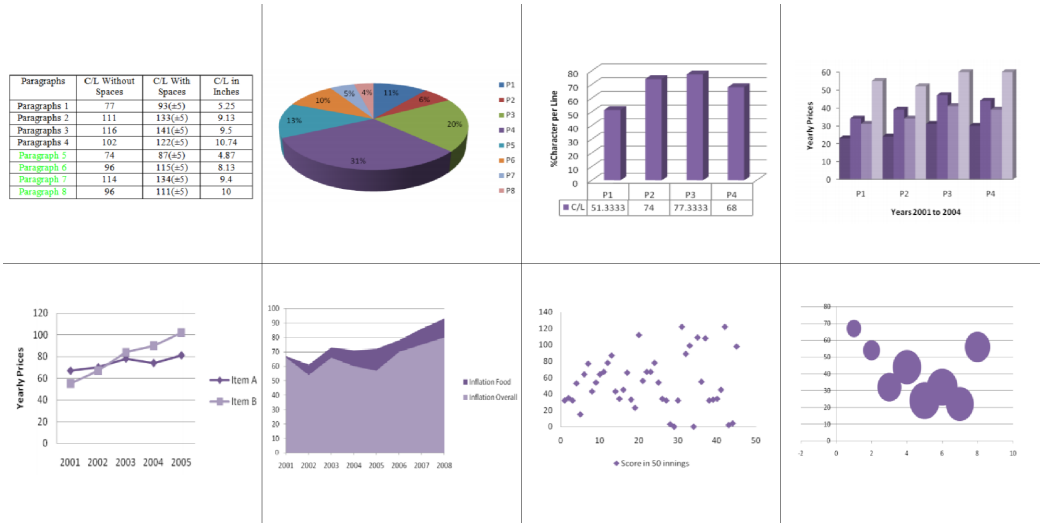

Figura 1.3: Da sinistra a destra, prima riga: tabella, grafico a torta, grafico a barre, istogramma; seconda riga: grafico a linee, grafico ad aree, grafico a dispersione, grafico a bolle.[6]

Nel caso i dati da rappresentare riguardino delle posizioni geografiche, è conveniente utilizzare come base una mappa. L'osservatore solitamente sa dove guardare per cercare l'area che gli interessa, il che rende i dati rapidamente capiti e assorbiti.[15]

Di seguito vengono riportate le più comuni tipologie di mappe [16]:

- **mappa coropletica**: è una mappa tematica dove le regioni geografiche sono colorate, ombreggiate o decorate rispetto a un valore. Questo tipo di mappa è utile per visualizzare il cambiamento di una variabile nelle

diverse regioni. Solitamente la tinta di colore identifica un attributo distinto, e il grado di luminosità misura la quantità di dati (le regioni meno popolate sono più chiare, altrimenti più scure).

- **mappa di calore**: una mappa di calore rappresenta l'intensità delle incidenze di un insieme di dati. Usa il colore per rappresentare l'intensità come la mappa coropletica, ma a differenza di quest'ultima non usa limiti geografici o geopolitici per raggruppare dati.

- **mappa di simboli proporzionali**: può rappresentare dati legati a specifici punti geografici o abbinati a una zona più ampia. In queste mappe, viene usato un simbolo per rappresentare il dato nel punto specifico o aggregato, e la sua dimensione geometrica ne misura il valore.

- **mappa di simboli graduati**: simile alla mappa di simboli proporzionali, ma la dimensione del simbolo è dipendente da un numero limitato di intervalli uniformi.[17]

- **mappa di distribuzione di punti**: indica il livello della presenza della variabile esaminata attraverso l'assegnazione di punti. Ciascun punto rappresenta la stessa unità di misura degli altri.

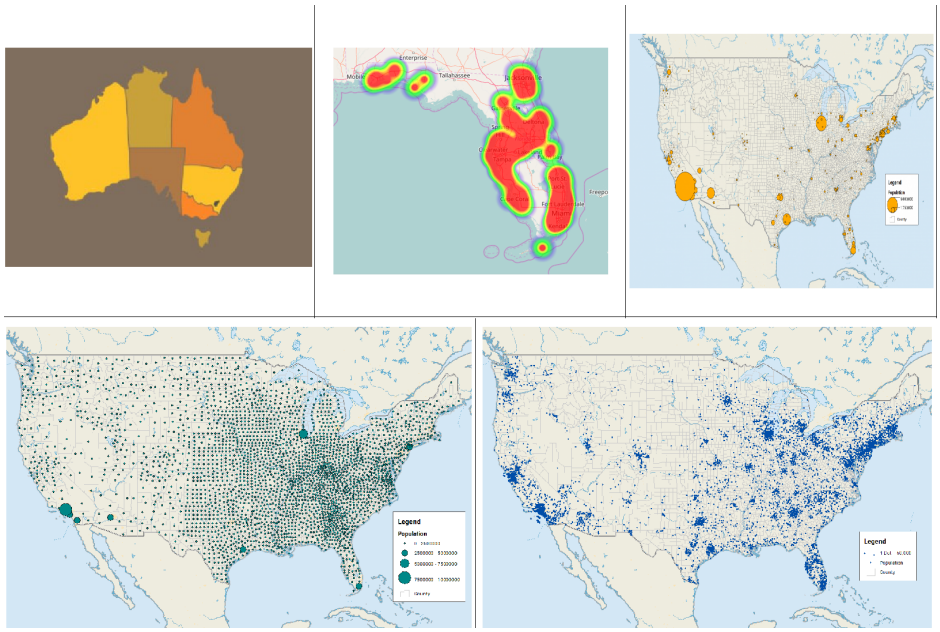

Figura 1.4: Da sinistra a destra, prima riga: mappa coropletica[18], mappa di calore[19], mappa di simboli proporzionali; seconda riga: mappa di simboli graduati, mappa di distribuzione di punti.[17]

1.3 Big data e open data

Si parla di **big data** quando la mole dei dati è talmente grande da rendere impossibile la loro archiviazione in un unico file e l'accessibilità dal software desiderato. La rappresentazione visuale dei big data è essenziale per individuare e comprendere errori di archiviazione.[15]

Discipline emergenti spesso entrano in disaccordo sulla definizione dei concetti di fondo. L'evoluzione rapida e caotica dei testi letterari sui big data ha impedito lo sviluppo di una definizione accettata in modo formale e universale.

L'assenza di una definizione consensuale dei big data ha spesso portato gli studiosi ad adottare "implicitamente" definizioni attraverso aneddoti, storie di successo, caratteristiche, aspetti tecnologici, mode o il loro impatto sulla società e nelle attività aziendali. Le definizioni di big data esistenti

forniscono prospettive molto diverse, denotando lo stato caotico dell'arte. È stato osservato che le varie definizioni possono essere classificate in quattro gruppi, dipendenti da dove è stata posta l'attenzione nella descrizione del fenomeno[20]:

- **attributi di dati**: secondo Laney i big data presentano tre proprietà, o dimensioni, che costituiscono il cosiddetto modello delle 3 V: la Velocità, la Varietà e il Volume. Per Velocità, si intende il tempo di attesa dell'elaborazione dei dati di un processo; con la tecnologia attuale, è possibile elaborarli in tempo reale. La Varietà consiste nella complessità dei formati con cui è possibile conservare i dati, da puri contenuti testuali a tipologie più complesse. Il Volume si riferisce alla quantità di dati generati ogni secondo.[21] Altri autori hanno esteso il modello delle 3 V con altre due caratteristiche: la Veridicità, che considera la varietà dei dati sorgente e la velocità alla quale tali dati possono variare, e il Valore, che si riferisce alla capacità di trasformare i dati in valore.[22]

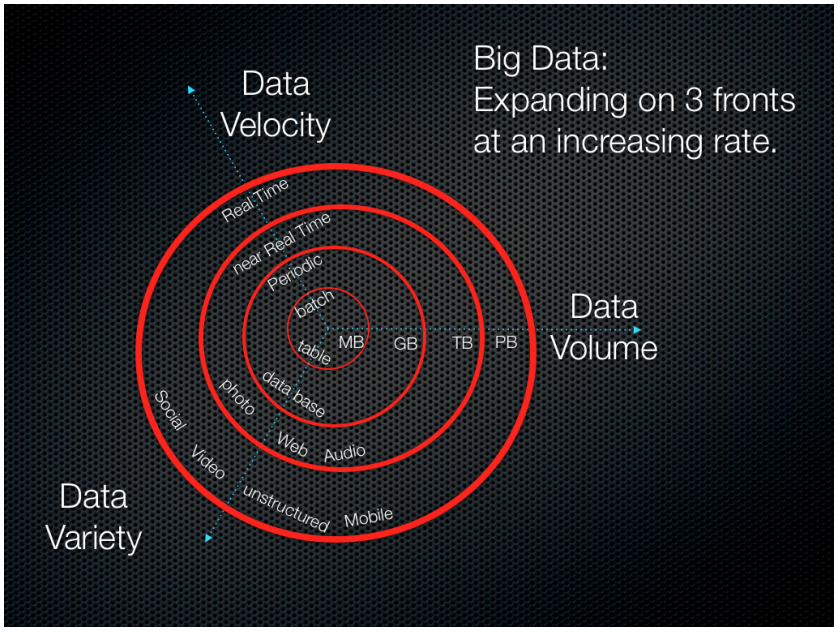

Figura 1.5: Grafico rappresentante il modello delle 3 V in forma grafica.[21]

- **bisogni tecnologici**: secondo Microsoft, "big data" descrive un processo in cui alta potenza di calcolo è applicata a insiemi di informazioni altamente complessi e di enormi dimensioni. Analogamente, il National Institute of Standards and Technology enfatizza il bisogno di un'architettura scalabile per conservazione, manipolazione e analisi efficienti.[20]

- **soglie**: alcune definizioni considerano i big data come superamenti di soglie. Dumbill sostiene che i dati possono essere definiti big data quando la capacità processuale dei sistemi di database convenzionali è superata e sono richiesti approcci alternativi. Fisher sostiene che il concetto di "big" in termini di dimensione è collegato alla legge di Moore, e di conseguenza alla capacità delle soluzioni di archiviazione commerciale.[20]

- **impatto sociale**: diverse definizioni mettono in rilievo l'effetto dell'avanzamento dei big data sulla società. Boyd e Crawford definiscono big data come un fenomeno sociale, tecnologico e culturale basato su tre elementi: tecnologia, analisi e mitologia, intese rispettivamente come la massimizzazione della potenza computazionale e della precisione algoritmica, l'identificazione di pattern in dataset di larga dimensione, e la convinzione che grandi gruppi di dati offrano una forma superiore di intelligenza. Mayer-Schönberger e Cukier descrivono big data in termini di tre movimenti principali di modalità di analisi di informazioni che migliorano la nostra comprensione di società e della sua organizzazione: più dati (utilizzo di tutti i dati disponibili piuttosto che soltanto di un campione), più disordinati (utilizzo di dati anche incompleti o incorretti), correlazione (viene data importanza al confronto tra le informazioni).[20]

L'analisi congiunta di queste definizioni esistenti permette di concludere che il concetto di big data può essere riassunto come un patrimonio infor-

mativo caratterizzato da alti tassi di Volume, Velocità e Varietà che richiede tecnologie e metodi analitici specifici per la sua trasformazione in Valore.

Prima dell'avvento dei big data, i database erano vincolati da tre attributi, dei quali erano disponibili solo due alla volta: grande e veloce, vario e veloce, grande e vario. Con l'avanzamento della potenza computazionale, nuove progettazioni di database e archiviazioni distribuite, tutti e tre possono essere realizzabili simultaneamente permettendo nuove di forme di analisi e fornendo punti di vista molto dettagliati di grandi sistemi in continua mutazione. Oltre al modello delle 3 V, la letteratura emergente indica altre caratteristiche chiave dei big data[23]:

- esaustivi nella portata, devono cercare di rappresentare tutto il sistema, o almeno campioni di dimensioni più grandi rispetto a quelli analizzati in studi di dati di minori dimensioni.
- con risoluzione granulata, con l'obiettivo di essere il più dettagliati possibile.
- relazionali in natura, contenenti campi comuni che rendano possibile l'unione con altri gruppi di dati.
- flessibili, mantenendo le proprietà di estindibilità e scalabilità.

Quando i big data sono di pubblico dominio, rientrano tra i cosiddetti **open data**, o **dati aperti**. La definizione ufficiale della Open Knowledge Foundation afferma che sono dati liberamente utilizzabili, riutilizzabili e ridistribuibili da chiunque, soggetti eventualmente alla necessità di citarne la fonte e di condividerli con lo stesso tipo di licenza con cui sono stati originariamente rilasciati.[24]

Date le spese e le risorse richieste per produrre insiemi di dati e il prezzo nel rivelare informazioni sul mondo, generalmente l'accesso tende a venire ristretto in qualche modo, ad esempio limitandolo a utenti approvati, richiedendo un pagamento o circoscrivendo le modalità di utilizzo dei dati attraverso licenze o politiche. Anche quando i dataset sono relativamente aperti e

accessibili, è possibile che siano necessari conoscenze e strumenti specialistici per poter dare loro un significato. Come conseguenza, i dati e le informazioni derivanti da essi sono di natura chiusa.

Le organizzazioni appartenenti al cosiddetto movimento dei dati aperti (Open Data movement) si pongono l'obiettivo di trasformare radicalmente questa situazione, rendendo accessibili dati per un ampio reimpiego e fornendo strumenti di ricerca di facile utilizzo che possano essere utilizzati da persone esterne al settore.

Il movimento si basa sui principi di apertura, partecipazione e collaborazione. Il suo obiettivo è democratizzare l'abilità di produrre informazione e conoscenza, piuttosto che confinare il potere dei dati ai rispettivi produttori e a chi è disposto a pagare per accedere a essi. In particolare, è stata posta attenzione sull'apertura dei dati prodotti da enti statali o da ricerche finanziate con fondi pubblici.[23]

Il punto fondamentale degli open data è il fatto che potenzialmente possono essere liberamente "mescolati" con dati provenienti da fonti anch'esse aperte. La capacità di diversi sistemi e organizzazioni di lavorare insieme è la chiave per realizzare il principale vantaggio pratico dell'apertura: aumenta in modo esponenziale la possibilità di combinare diverse basi di dati, e quindi sviluppare nuovi e migliori prodotti e servizi.[25]

1.3.1 Licenze

Per rendere dei dati aperti, è necessaria l'applicazione di una licenza aperta prima della loro pubblicazione. In diverse giurisdizioni esistono diritti di proprietà dei dati che impediscono a terze parti di usarli, riutilizzarli e distribuirli senza un permesso esplicito. Anche in luoghi dove l'esistenza di diritti non è certa, è importante applicare una licenza semplicemente per motivi di chiarezza. Per questo motivo, è opportuno assegnare una licenza ai dati che si vogliono rendere aperti.[26]

Le più importanti licenze per la libera circolazione di materiale creativo diverso dal software, sono quelle nate in seno all'organizzazione non profit

Creative Commons. Tali licenze nascono dall'esigenza di permettere la libera circolazione del materiale creativo protetto dal diritto d'autore dal momento che sono ispirate al modello "alcuni diritti riservati". [27]

Le licenze Creative Commons si strutturano idealmente in due parti[27]: una prima parte indica quali sono le libertà che l'autore vuole concedere alla sua opera ed una seconda parte che chiarisce a quali condizioni è possibile utilizzare la stessa.

In generale, si può affermare che la prima parte delle licenze indica le libertà concesse dall'autore per la propria opera relative alla possibilità di condividerla e rielaborarla, mentre la seconda prevede l'individuazione delle condizioni per l'utilizzo dell'opera stessa attraverso quattro clausole, che possono essere agevolmente utilizzate creando delle vere e proprie combinazioni di diritti:

- **Attribuzione**: questa clausola impone che si debba riconoscere la paternità dell'opera all'autore originario. Si tratta di una clausola sempre presente in tutte le tipologie di licenze Creative Commons e con la stessa viene imposto di segnalare sempre la fonte.

- **Non commerciale**: tale clausola impone che il riutilizzo dell'opera non possa essere consentito per fini commerciali. Tuttavia, occorre precisarne la portata: infatti, essa indica che se si distribuiscono copie dell'opera, non si può farlo in una maniera tale che sia prevalentemente perseguito un vantaggio commerciale o un compenso monetario. Per utilizzare in tal senso il materiale distribuito, è necessario chiedere uno specifico consenso all'autore.

- **Non opere derivate**. L'applicazione di tale clausola indica l'impossibilità di trasformare, alterare o modificare l'opera. Anche in tal caso, come accade per la clausola non commerciale, qualora si volessero realizzare opere derivate sarebbe necessario ottenere uno specifico permesso da parte dell'autore originario.

- **Condividi allo stesso modo**. È anche conosciuta come clausola virale della licenza (tecnicamente clausola di persistenza). Infatti, se applicata stabilisce che l'alterazione, trasformazione o sviluppo dell'opera, obbliga e redistribuire l'opera risultante soltanto per mezzo di una licenza identica a quella attribuita all'opera originaria. Tale clausola garantisce che le libertà concesse dall'autore, siano attribuite anche alle opere derivate.

  Soltanto le licenze "Attribuzione" sono compatibili con il principio dell'Open by default, per il quale i dati e i documenti pubblicati dalle Pubbliche Amministrazioni senza una esplicita licenza d'uso che ne definisca le possibilità e i limiti di riutilizzo sono da intendersi come dati aperti.

Le licenze Creative Commons sono nate con riferimento a testi, immagini, video e musica, ma non sono particolarmente adatte per le banche dati.[27] Per queste ultime sono state dunque sviluppate apposite licenze, come la ODbL (Open Database Licence), creata nell'ambito del progetto della Open Knowledge Foundation volto ad affermare l'uso di licenze "aperte" anche per i database. Inizialmente pensata per il progetto relativo ai dati cartografici OpenStreetMap, è diventata un punto di riferimento anche in ambito pubblico. La ODbL, offre all'utente tre diritti fondamentali:

- To share: tale diritto consiste nell'offrire la possibilità di copiare, distribuire ed utilizzare il database.

- To create: l'utilizzo della licenza OdbL garantisce la possibilità di lavorare e creare nuove opere a partire dal database fornito.

- To adapt: è possibile modificare, trasformare e costruire opere derivate a partire dall'iniziale database. Come si è visto anche per licenze Creative Commons e come accade per tutte le licenze di tipo aperto, i diritti attribuiti dalla OdbL richiedono di rispettare alcuni principi:

- Attribute: occorre sempre rendere possibile e garantire l'uso del database o delle opere da esso derivate secondo i termini della licenza OdbL.
- Share-Alike: l'uso di versioni adattate del database, nonché la creazione e distribuzione di database derivati o adattati, deve essere effettuata sempre nel rispetto dei termini della licenza OdbL. Si tratta di una clausola molto simile, se non del tutto identica, alla clausola «condividi allo stesso modo» delle licenze Creative Commons.
- Keep open: se si redistribuisce il database o una sua versione adattata, è necessario non utilizzare sistemi che ne limitino l'uso.

L'Università di Bologna è il primo ateneo in Italia a dotarsi di un portale che possa raccogliere, organizzare e mettere a disposizione in rete pacchetti di dati indicizzati e navigabili, distribuiti con licenza Creative Commons.[28]

Capitolo 2

Tecnologie utilizzate

Questo capitolo ha lo scopo di descrivere sinteticamente le principali tecnologie utilizzate nello sviluppo del progetto.

2.1 HTML

HTML (**HyperText Markup Language**, tradotto letteralmente linguaggio a marcatori per ipertesti) è un linguaggio di markup ideato per creare siti web. È in costante revisione ed evoluzione sotto la direzione dell'organizzazione World Wide Web Consortium (W3C).

Per "ipertesto" si intende un testo che contiene hyperlink, collegamenti ipertestuali che conducono ad altri testi. La navigazione attraverso gli ipertesti non è lineare, ma è possibile accedere alle varie pagine senza un ordine prefissato. I marcatori corrispondono invece ai cosiddetti tag HTML, che assegnano delle determinate caratteristiche al testo e agli altri elementi della pagina.[29]

Sebbene l'HTML supporti l'inserimento di script e oggetti esterni quali immagini o filmati, non è un linguaggio di programmazione: non prevedendo alcuna definizione di variabili, strutture dati, funzioni o strutture di controllo che possano realizzare programmi, il suo codice è in grado soltanto di strutturare e decorare dati testuali.[30]

I file HTML possono essere visualizzati attraverso un web browser, come Google Chrome o Mozilla Firefox. Il browser legge il file e converte il testo in una forma visibile attraversando due fasi: parsing e rendering. Durante la fase di parsing, il browser legge i markup nel documento, li suddivide in componenti, e costruisce il relativo Document Object Model (DOM), una forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti. A costruzione dell'albero DOM ultimata e caricamento dei fogli di stile CSS caricati e analizzati, il browser entra nella fase di rendering: ogni nodo dell'albero DOM verrà renderizzato e mostrato nel browser.

Ciascun browser renderizza i file HTML in modi leggermente diversi tra loro; durante lo sviluppo di pagine web, è quindi importante effettuare dei test su più browser possibili.[31]

Attualmente l'ultima versione di HTML disponibile è HTML5. È il risultato di una collaborazione tra W3C e il gruppo Web Hypertext Application Technology Working Group (WHATWG); le organizzazioni hanno a iniziato a collaborare nel 2006 per ridurre la dipendenza da plugin, migliorare la gestione degli errori e sostituire script con più marcatori. Le differenze più significative dalle precedenti versioni sono il supporto nativo di file multimediali e la possibilità di conservare dati localmente nel browser senza utilizzare cookies.[32]

2.2 CSS

CSS sta per **Cascading Style Sheets**, letteralmente fogli di stile a cascata. Mentre HTML è utilizzato per strutturare un documento web definendo elementi testuali come titoli e paragrafi e allegando immagini, video e altri file multimediali, CSS si occupa di personalizzare lo stile degli elementi HTML contenuti nel documento (layout di pagina, colori, font).[33]

Uno stile è composto da due parti: il selettore, ovvero gli elementi che il browser formatta, e le relative istruzioni di formattazione contenute in un

blocco. I selettori più comuni sono[34]:

- **di tipo**: usano i nomi degli elementi a cui si riferiscono (`p` , `span` , `input` , ecc.).
- **di classe**: coinvolgono gli elementi che hanno come attributo `class` uno dei valori specificati (`.active` , `.container` , `.h-100` , ecc.).
- **di ID**: coinvolgono gli elementi che hanno come attributo `id` il valore specificato (`#first` , `#main` , `#footer` , ecc.). Ciascun ID deve essere univoco nel documento.
- **universali**: seleziona tutti i nodi, relativi a un solo namespace, o in tutti i namespace (`*` `ns|*` `*|*`).
- **di attributo**: scelgono i nodi in base al valore di uno dei loro attributi (`[attr]` , `[attr=value]` , `[attr =value]` , `[attr|=value]` , `[attr^=value]` , `[attr$=value]` , `[attr*=value]`)

Ciascuna dichiarazione è composta da una proprietà e un valore; una proprietà è una parola che descrive un determinato effetto di stile, e il rispettivo valore ne determina le caratteristiche. Nell'esempio sottostante, vengono assegnati il valore `red` alla proprietà `color` e il valore `1.5em` alla proprietà `font-size` in tutti gli elementi marcati con il tag `<p>` ; in altre parole, lo stile rende tutti i paragrafi di colore rosso e alti 1.5em (3/2 rispetto all'altezza di default fissata dal browser).[35]

```
p {
 color: red;
 font-size: 1.5em;
}
```

Come per il linguaggio HTML, è possibile scrivere codice CSS attraverso un editor o elaboratore di testo, ed esistono tre modalità per applicarlo alle pagine HTML:

- **stile inline**: le proprietà CSS sono contenute nell'attributo `style` del tag relativo all'elemento HTML interessato.

- **stile interno**: è possibile specificare un insieme di regole CSS nella sezione `head` di un file HTML; esse interessano unicamente gli elementi HTML descritti nel file relativo.

- **stile esterno**: i fogli di stile esterni sono salvati in file con formato `.css` e sono collegati al documento HTML attraverso il tag `link`.

Gli stili interni hanno la priorità sugli stili esterni, e gli stili inline hanno la priorità sugli altri due: ciò significa che se la stessa proprietà di un elemento è definita con modi diversi, verranno applicate sovrascritture di stile in base all'ordine di priorità.[36]

2.2.1 CSS Modules

I **CSS Modules** sono file CSS in cui tutti i nomi di classi e animazioni hanno la visibilità (scope) locale di default.

Non sono una specifica ufficiale o un'implementazione del browser ma un processo nella fase di build che modifica nomi di classi e selettori al fine di modificare il loro scope.[37]

Con CSS Modules, i nomi delle classi CSS diventano paragonabili a variabili locali di JavaScript. Un CSS Module in sé è un semplice file `.css`, ma viene chiamato in questo modo quando viene utilizzato un compilatore apposito.[38]

I markup devono essere scritti in un file JavaScript, come nel seguente esempio:

```
1 import styles from "./styles.css";
2
3 element.innerHTML =
4   `<h1 class="${styles.title}">
5      An example heading
6    </h1>`;
```

Durante la fase di build, il compilatore accede al file `styles.css` importato, dopodiché analizza il file JavaScript e rende la classe `.title` accessibile

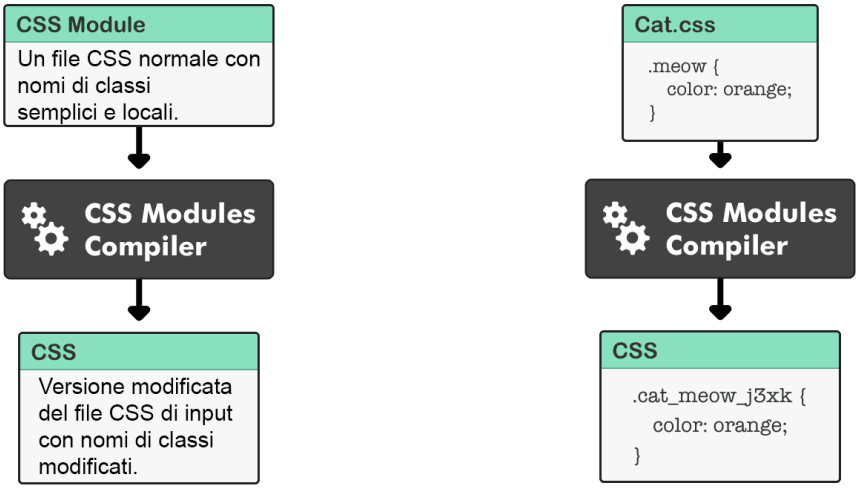

Figura 2.1: Rappresentazione visuale della compilazione di CSS Modules con esempio.[38]

attraverso `styles.title`. Il compilatore crea dunque nuovi e separati file HTML e CSS, sostituendo sia la classe HTML che il selettore CSS con nuove stringhe di caratteri (ad esempio `_styles__title_309571057`).

Questo approccio è stato progettato per risolvere il problema della visibilità globale di CSS e poter applicare un insieme di stili con certezza a un singolo componente.[37]

2.2.2 Sass

Sass è un pre-processore dei CSS, compatibile al 100% con la sintassi di quest'ultimi, che permette di lavorare con maggior semplicità e velocità sugli stessi.[39]

Sono disponibili due sintassi per Sass. La prima, conosciuta come **SCSS** (Sassy CSS), è un'estensione della sintassi di CSS. Ciò significa che ogni foglio di stile CSS valido è anche un file SCSS valido con lo stesso significato. La seconda e più vecchia sintassi, conosciuta come la sintassi indentata (o semplicemente "Sass") fornisce un modo più conciso per scrivere CSS.

Usa l'indentazione piuttosto che parentesi graffe per indicare l'annidamento di selettori, e andate a capo piuttosto che punti e virgola per separare le proprietà.[40]

Questo progetto ha fatto uso della sintassi SCSS.

2.2.3 Bootstrap

Bootstrap è un framework front-end progettato per avviare lo sviluppo front-end di applicazioni web e siti. Include HTML e CSS di base per tipografia, icone, forme, bottoni, tabelle, layout a griglia e navigazione, oltre a plugin di JQuery personalizzati e supporto per i layout responsive.

È stato testato e supportato dai principali browser moderni. A partire dalla versione 2.0, i pacchetti di download di base contengono file JavaScript e CSS cumulativi.[41]

2.3 JavaScript

JavaScript è un linguaggio ad alto livello, dinamico, multi-paradigma, basato sui prototipi, debolmente tipizzato e interpretato, comunemente usato per scripting lato client nei web browser e conforme alla specifica ECMAScript. Può essere eseguito anche esternamente al browser attraverso framework come Node.js o Google Apps Script. Nonostante il nome, è indipendente dal linguaggio Java e condivide solo somiglianze superficiali.

JavaScript funziona su quasi tutti i sistemi operativi, e i browser principali includono un motore apposito. È tipicamente usato per manipolare DOM e CSS all'interno del browser, in modo da rendere possibili scripting d'interfaccia, animazioni, automazione, validazione lato client e molto altro.[42]

2.3.1 React

React, conosciuto anche come **ReactJS**, è una libreria JavaScript progettata per creare interfacce utente, strutturate in componenti riutilizzabili in diversi punti dell'applicazione.[43]

È stata creata da Jordan Walke, un ingegnere del software che lavora per Facebook. Facebook ne fa uso dal 2011, mentre Instagram dal 2012.

React permette agli sviluppatori di creare applicazioni web di grandi dimensioni che possono cambiare dati senza ricaricare la pagina. Lo scopo principale di React è essere veloce, scalabile e semplice. Corrisponde alla "view" del template Model Control View (MVC). Può essere utilizzata in combinazione ad altre librerie o framework JavaScript, come AngularJS.[44]

I componenti React implementano un metodo `render()` che riceve dati in input e ritorna cosa deve visualizzare. I dati passati in input al componente possono essere acceduti da `render()` via `this.props`.

Oltre a ricevere dati in input, un componente può mantenere i dati del suo stato interno (accessibili via `this.state`). Quando lo stato di un componente cambia, il codice markup generato viene aggiornato automaticamente invocando di nuovo `render()`.[43]

2.3.2 Node.js

Node.js è un runtime JavaScript basato sugli eventi, non bloccante e asincrono costruito sul motore JavaScript V8 di Chrome e sulla libreria libuv. È usato per sviluppare applicazioni che fanno un ampio utilizzo di JavaScript sia lato client che lato server, e che quindi beneficiano della riciclabilità del codice e dell'assenza di commutazione di contesto.

Se un task entra in fase di stallo o pausa per l'esecuzione di un operazione I/O, può essere avviato un altro task. Ciò garantisce un alto tasso di efficienza, in quanto non è necessario attendere l'esecuzione di un singolo task per proseguire l'esecuzione dell'intero programma.

Per facilitare lo sviluppo di codice JavaScript complesso, Node.js supporta lo standard CommonJS che permette uno sviluppo modularizzato e la distribuzione di software in pacchetti attraverso il Node Package Manager (NPM).[45]

Node.js usa un `[event loop][]` come costrutto di runtime invece che una libreria come in altri linguaggi. In altri sistemi, c'è sempre una chiamata bloccante per avviare l'event-loop. In genere il comportamento è definito tramite callback all'inizio di uno script e alla fine avvia un server attraverso una chiamata bloccante come EventMachine::run(). In Node.js non esiste alcuna chiamata per avviare il ciclo. Node.js entra semplicemente nel ciclo degli eventi dopo aver eseguito lo script di input, e ne esce quando non ci sono più callback da eseguire.[46]

Quando Node.js viene avviato, inizializza l'event loop, processa gli script di input forniti che potrebbero effettuare chiamate API asincrone, programmare timer o chiamare `process.nextTick()`, per poi processare l'event loop.[47]

Le operazioni dell'event loop possono essere suddivise in fasi; ciascuna di esse ha una coda FIFO di callback da eseguire. Quando l'event loop entra in una fase, eseguirà qualsiasi operazione a essa relativa, per poi eseguire callback nella coda di quella fase finché non viene svuotata o quando è stato raggiunto il numero massimo consentito di callback eseguite, e quindi entrare nella fase successiva.[47]

Le fasi dell'event loop sono le seguenti:

- **timer**: questa fase esegue le callback programmate da `setTimeout()` e `setInterval()`.

- **callback in attesa**: esegue le callback di input/output prelevate dalla coda di attesa relative a operazioni di sistema, come ad esempio errori TCP.

- **inattività, preperazione**: operazioni interne.

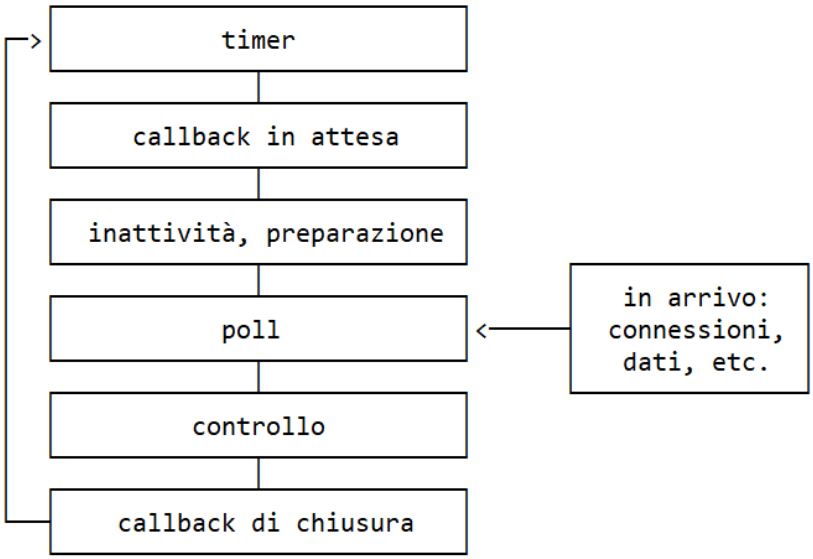

Figura 2.2: Rappresentazione visuale delle transizioni delle fasi dell'event loop.[47]

- **poll**: vengono prelevati e inseriti nella coda nuovi eventi collegati al completamento di operazioni di I/O. Se la coda non è vuota, verranno processate le diverse callback in essa presenti. In caso contrario, se sono state registrate delle callback con la funzione `setImmediate()`, verrà terminata questa fase e si passerà alla fase successiva per processarle. Nel caso in cui non siano state registrate callback con la funzione `setImmediate()`, l'event loop resta in attesa che vengano aggiunte nuove callback alla coda. Mentre è in questa fase, l'event loop controlla se sono presenti callback che devono essere eseguite nella coda dei timer e, se lo sono, ritorna nella prima fase ed esegue le callback presenti in quella coda.[48]

- **controllo**: vengono invocate le callback `setImmediate()`.

- **callback di chiusura**: vengono gestiti gli eventi di chiusura, per pulire lo stato dell'applicazione.

La funzione `process.nextTick()` viene eseguita subito dopo il completamento dell'operazione corrente, indipendentemente dalla fase in cui l'event loop si trova. È principalmente utilizzata per gestire errori, risorse non necessarie, ritentare richieste non andate a buon fine, oppure per effettuare operazioni dopo l'esaurimento dello stack delle chiamate ma prima che l'event loop continui.[47]

2.3.3 Koa.js

Koa.js è un framework per applicazioni web Node.js, progettato dagli sviluppatori di **Express.js**, software della stessa categoria.[49] La differenza principale tra i due framework è l'approccio all'utilizzo dei middleware, funzioni che hanno accesso all'oggetto richiesta (`req`), all'oggetto risposta (`res`) e al middleware da eseguire successivamente (`next`).[50]

Il modulo Koa principale include solamente il middleware kernel. Express comprende un framework completo, con funzionalità come routing e template che in Koa sono moduli separati; ciò rende Koa più modulare, in quanto è sufficiente includere solo i moduli di cui si vuole fare uso.

Express estende gli oggetti `req` e `res` di Node, mentre Koa li sostituisce completamente con un oggetto Context `ctx` , tra le cui proprietà figurano `ctx.request` e `ctx.response` , e che fornisce diversi metodi utili per scrivere applicazioni web e API. Viene creato un Context a ogni richiesta, e molte delle sue proprietà e metodi sono delegati ai suoi `ctx.request` e `ctx.response` ; ad esempio, `ctx.body` e `ctx.status` delegano all'oggetto `response` , e `ctx.path` e `ctx.method` all'oggetto `request` .[49] Koa è progettato per migliorare l'esperienza complessiva della scrittura di routine di middleware, utilizzando nuove caratteristiche di Node: `async` / `await` .[51]

La dichiarazione di una funzione è preceduta dalla parola chiave `async` quando è prevista la restituzione di una Promise, un oggetto che rappresenta un'operazione ancora non completata ma che lo sarà in futuro[52]; nel corpo delle funzioni precedute da `async` , viene utilizzata la parola chiave `await`

prima della chiamata di una Promise per attendere il suo risultato prima di proseguire con l'esecuzione del resto della funzione.[53]

In JavaScript, le chiamate di rete sono sempre asincrone. Vi sono diversi modi per scrivere codice asincrono: callback, Promise e la nuova sintassi `async` / `await` introdotta in Node 7.6. Il codice seguente mostra un esempio di utilizzo di callback:

```
function myFunction(params, callback){
  // chiamata asincrona
  asyncCall(params, function(res) {
    callback(res);
  })
}

myFunction(myParams, function(data){
  // operazione che utilizza data
})
```

La funzione `myFunction()` prende come secondo parametro un dato di tipo funzione. L'implementazione interna di myFunction porterebbe tipicamente all'esecuzione di una chiamata asincrona; potrebbe essere una chiamata API, o un'interrogazione al database. Al completamento dell'operazione asincrona, `myFunction()` chiamerà la funzione passata, e la routine di chiamata può accedere alla risposta.

Questo metodo è ragionevole quando si tratta di una singola chiamata, ma nel caso sia necessario creare molteplici chiamate asincrone collegate tra loro, il codice risultante sarà difficile da leggere e mantenere. Questo problema è conosciuto come "callback hell" (letteralmente "inferno di callback").

Utilizzando `async` / `await`, il codice assume questo aspetto:

```
async myFunction(){
  try {
    let res = await asyncCall();
    return res;
  }
```

```
6   catch(err){
7     console.log("Error: " + err);
8   }
9 }
10
11 let result = await myFunction();
```

Il risultato è più corto e facile da interpretare. Il codice viene eseguito in modo asincrono, ma ha l'aspetto di codice sincrono.[51]

2.4 CSV

Il **comma-separated values** (abbreviato in **CSV**) è un formato di file basato su file di testo utilizzato per l'importazione ed esportazione di una tabella di dati. Non esiste uno standard formale che lo definisca, ma solo alcune prassi più o meno consolidate.

In questo formato, ogni riga della tabella (o record della base dati) è normalmente rappresentata da una linea di testo, che a sua volta è divisa in campi (le singole colonne) separati da un apposito carattere separatore, ciascuno dei quali rappresenta un valore.

Il formato CSV non specifica una codifica di caratteri, né la convenzione per indicare il fine linea, né il carattere da usare come separatore tra campi e nemmeno convenzioni per rappresentare date o numeri e se la prima riga è solo di intestazione o meno. Questi dettagli possono dover essere specificati dall'utente tutte le volte che si importano o esportano dati in formato CSV in un programma.[54]

2.5 JSON

JSON (**JavaScript Object Notation**) è un semplice formato per lo scambio di dati. Per le persone è facile da leggere e scrivere, mentre per le macchine risulta facile da generare e analizzarne la sintassi. Si basa su

un sottoinsieme del Linguaggio di Programmazione JavaScript, Standard ECMA-262 Terza Edizione - Dicembre 1999.

JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C, C++, C#, Java, JavaScript, Perl, Python, e molti altri. Questa caratteristica fa di JSON un linguaggio ideale per lo scambio di dati.

JSON è basato su due strutture:

- Un insieme di coppie nome/valore: in diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.

- Un elenco ordinato di valori: nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.

Queste sono strutture di dati universali. Virtualmente tutti i linguaggi di programmazione moderni li supportano in entrambe le forme. È sensato che un formato di dati che è interscambiabile con linguaggi di programmazione debba essere basato su queste strutture.

In JSON, assumono queste forme:

- un **oggetto** è una serie non ordinata di nomi/valori. Un oggetto inizia con una parentesi graffa sinistra `{` e finisce con una parentesi graffa destra `}`. Ogni nome è seguito da due punti `:` e le coppie di nome/valore sono separate da virgole `,`.

- un **array** è una raccolta ordinata di valori. Un array comincia con una parentesi quadra sinistra `[` e finisce con una parentesi quadra destra `]`. I valori sono separati da una virgola `,`.

- un **valore** può essere una stringa tra virgolette, un numero, un booleano (`true` / `false`) o `null`, un oggetto, oppure un array. Queste strutture possono essere annidate.

- una **stringa** è una raccolta di zero o più caratteri Unicode, tra virgolette; per le sequenze di escape utilizza la barra rovesciata. Un singolo carattere è rappresentato come una stringa di caratteri di lunghezza uno. Una stringa è molto simile ad una stringa C o Java.
- un **numero** è molto simile ad un numero C o Java, a parte il fatto che i formati ottali e esadecimali non sono utilizzati.
- i **caratteri di spaziatura** possono essere inseriti in mezzo a qualsiasi coppia di token.

A parte alcuni dettagli di codifica, questo descrive totalmente il linguaggio.[55]

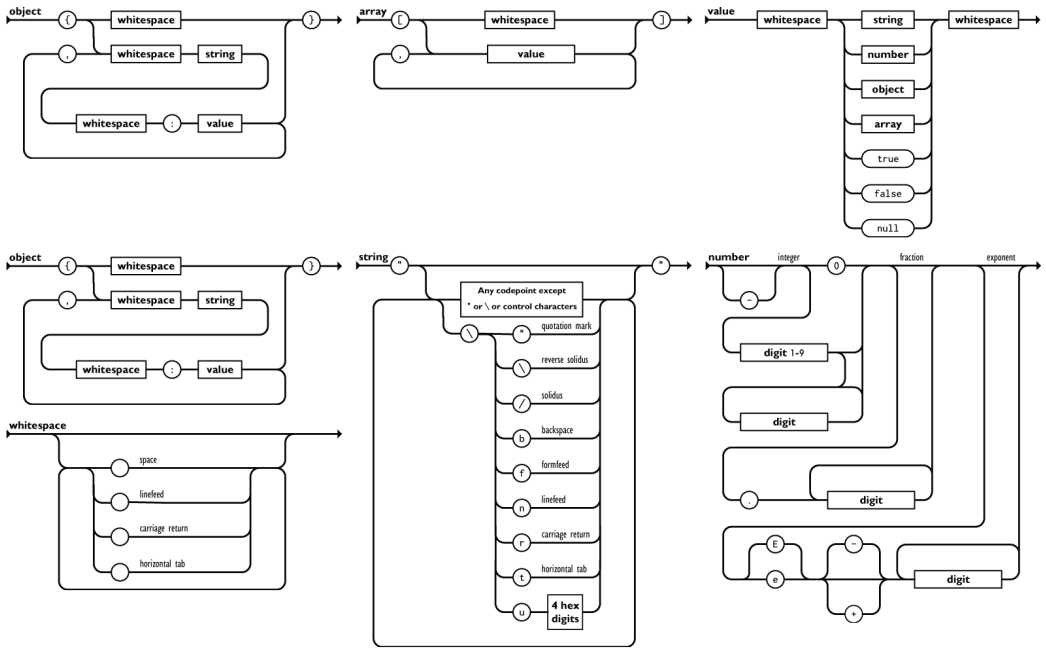

Figura 2.3: Rappresentazioni visuali delle strutture JSON.[55]

2.5.1 GeoJSON

GeoJSON è un formato utilizzato per codificare diverse tipologie di strutture di dati geografici. Supporta punti, linee, poligoni e collezioni multiple di queste tipologie. Oggetti geometrici con proprietà aggiuntive sono oggetti `Feature`; insiemi di `Feature` sono contenuti in oggetti `FeatureCollection`.

Un esempio di `Feature` è riportato nel seguente codice:

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

GeoJSON è stato standardizzato nel 2015 dalla Internet Engineering Task Force (IETF); la specifica standard più recente è stata pubblicata in agosto 2016 ed è denominata RFC 7946.[56]

2.6 MongoDB

MongoDB è un DBMS non relazionale open source che utilizza un modello orientato ai documenti, il quale supporta diverse forme di dati. È una delle numerose tecnologie di database non relazionali nate a metà degli anni 2000 sotto il nome di NoSQL per l'utilizzo in applicazioni per big data e altre operazioni di elaborazione di dati che non beneficiano dell'utilizzo di modelli relazionali rigidi. L'architettura MongoDB è composta da collezioni e documenti, piuttosto che da tabelle e righe.

In MongoDB, un record è un documento, che consiste in una struttura dati composta da coppie di campo e valore. I documenti MongoDB usano una variante degli oggetti JSON chiamata Binary JSON (BSON), che può

contenere ulteriori tipologie di dati. I campi dei documenti sono analoghi alle colonne dei database relazionali, e i valori in essi contenuti possono essere una varietà di tipi di dati, tra cui altri documenti, array e array di documenti.

I documenti, forniti di una chiave primaria come identificatore univoco, sono l'unità di base dei dati in MongoDB. Le collezioni contengono insiemi di documenti, e sono analoghe alle tabelle nei database relazionali. Le collezioni possono contenere qualsiasi tipo di dato, ma come restrizione i dati in una collezione non possono essere sparsi in altri database.

La mongo shell è un'interfaccia JavaScript a MongoDB interattiva che permette di effettuare interrogazioni e aggiornamenti sui dati, oltre ad eseguire operazioni di amministrazione. La shell è un componente standard delle distribuzioni open source di MongoDB. Una volta installato MongoDB, gli utenti connettono la mongo shell alle istanze MongoDB in esecuzione.

Il formato BSON con cui sono conservati i documenti permette di rappresentare i dati in forma binaria; ciò rende possibile lo sharding automatico, una proprietà chiave di MongoDB che permette alle collezioni di essere distribuite su molteplici sistemi a seguito della crescita del volume dei dati da conservare.

A differenza di altri database NoSQL, MongoDB non richiede l'utilizzo di schemi di database (strutture logiche dei dati contenuti nel database) predefiniti e memorizza qualsiasi tipo di dato. Ciò dà agli utenti la flessibilità di creare un qualsiasi numeri di campi in un documento, facilitando la scalabilità dei database MongoDB rispetto a quelli relazionali.

Il fatto di poter disporre di un oggetto JSON-like permette una rapida manipolazione delle informazioni, senza necessitare di operazioni di join e quindi ridurre il costo delle operazioni.[57]

2.6.1 Mongoose

Mongoose è una libreria di Object Data Modelling (ODM) per MongoDB e Node.js. Gestisce le relazioni tra i dati, permette la definizione di

schemi, ed è utilizzata come convertitore tra oggetti nel codice e la loro rappresentazione in MongoDB.

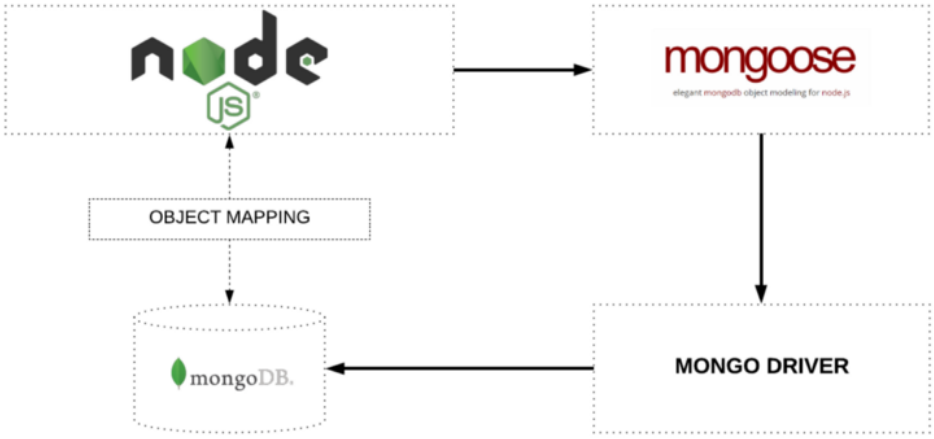

Figura 2.4: Mappatura di oggetti tra Node e MongoDB tramite Mongoose.[58]

La chiamata `require("mongoose")` restituisce un oggetto Singleton, così come avviene per qualsiasi modulo importato in ES6. Ciò significa che alla prima chiamata verrà creata e restituita un'istanza della classe `Mongoose`, e alle chiamate successive verrà restituita la stessa istanza creata precedentemente e restituita la prima volta.[58]

Uno schema descrive il costrutto dei dati di un documento, ovvero definisce il nome e il tipo di ciascun elemento. È ragionevole associare uno schema a ciascuna collezione presente nel database. Un esempio di schema in Mongoose è il seguente:

```
var userSchema = new mongoose.Schema({
 name: String,
 email: String,
 createdOn: Date,
 verified: Boolean
});
```

Un altro elemento chiave di Mongoose è il cosiddetto modello. Esso è la versione compilata dello schema; un'istanza del modello verrà mappata a un documento nel database. Si occupa di gestire la lettura, la creazione, l'aggiornamento e la cancellazione dei documenti.[59]

```
var User = mongoose.model("User", userSchema);
```

Le specifiche di connessione tra Mongoose e il database sono contenute nella cosiddetta stringa di connessione, espressa nel seguente formato:[60]

```
mongodb://[username:password@]host1[:port1][,...hostN[:portN
    ]][/[database][?opzioni]]
```

Vi sono due metodi per effettuare la connessione al database con Mongoose: `mongoose.connect` e `createConnection`.[61]

Il primo imposta la connessione di default, la quale sarà accessibile da qualsiasi punto dell'applicazione se impostata correttamente.

```
var dbURI = "mongodb://localhost/mydatabase";
mongoose.connect(dbURI);
```

Il secondo viene utilizzato nel caso sia necessario stabilire più di una connessione, che sia allo stesso database che a un altro.

```
var dbURI = "mongodb://localhost/myadmindatabase";
var adminConnection = mongoose.createConnection(dbURI);
```

2.7 Python

Python è un linguaggio di programmazione interpretato, interattivo, orientato agli oggetti, dinamico e fortemente tipizzato che è utilizzato per un vasta gamma di applicazioni. Incorpora moduli, eccezioni, tipizzazione dinamica, tipi di dati dinamici ad alto livello e classi. È provvisto di interfacce a diverse chiamate di sistema e librerie, così come a vari sistemi a finestre; è estendibile in C o C++ ed è un linguaggio portabile: è compatibile con Mac, diverse varianti Unix e i sistemi Windows da Windows 2000 in poi.

Il linguaggio è appoggiato da una vasta libreria standard che copre diverse funzionalità, come l'elaborazione di stringhe (espressioni regolari, Unicode, calcolo di differenze tra file), protocolli internet (HTTP, FTP, SMTP, XML-RPC, POP, IMAP, programmazione CGI), ingegneria del software (testing di unità, logging, profilazione, parsing di codice Python) e interfacce di sistemi operativi (chiamate di sistema, file system, socket TCP/IP).

Python permette ai programmatori di esprimere concetti in meno linee di codice rispetto a molti altri linguaggi, come ad esempio il C, ed è provvisto di costrutti progettati con l'intento di essere utilizzati per creare programmi chiari in diversi ambiti.

È stato creato dal programmatore Guido van Rossum e rilasciato nel 1991; le versioni successive del linguaggio, Python 2 e Python 3, sono state pubblicate rispettivamente il 16 ottobre 2000 e il 3 dicembre 2008.

Come altri linguaggi dinamici, Python è spesso usato come linguaggio di scripting.[62]

2.7.1 GeoPy

GeoPy è un programma client di Python 2 e 3 per diversi servizi web di geocodifica.

GeoPy facilita agli sviluppatori Python la localizzazione delle coordinate di indirizzi, città, stati e punti di riferimento in tutto il mondo, utilizzando geocodificatori di terze parti e altre fonti di dati.

Ogni servizio di localizzazione disponibile, come Google Maps, Bing Maps o Nominatim, ha una propria classe in `geopy.geocoders` che astrae l'API del servizio. Ciascun geocodificatore definisce almeno un metodo `geocode`, per individuare una posizione data una stringa, e un metodo `reverse`, che converte un paio di coordinate in un indirizzo. Ciascun geocodificatore accetta le credenziali o le impostazioni necessarie per interagire con il suo servizio durante la sua inizializzazione.

Di seguito viene riportato un esempio di utilizzo dei due metodi precedentemente citati.[63]

```
>>> from geopy.geocoders import Nominatim
>>> geolocator = Nominatim(user_agent="
    specify_your_app_name_here")
>>> location = geolocator.geocode("175 5th Avenue NYC")
>>> print(location.address)
Flatiron Building, 175, 5th Avenue, Flatiron, New York, NYC,
    New York, ...
>>> print((location.latitude, location.longitude))
(40.7410861, -73.9896297241625)
>>> print(location.raw)
{'place_id': '9167009604', 'type': 'attraction', ...}

>>> location2 = geolocator.reverse("52.509669, 13.376294")
>>> print(location2.address)
Potsdamer Platz, Mitte, Berlin, 10117, Deutschland, European
    Union
>>> print((location2.latitude, location2.longitude))
(52.5094982, 13.3765983)
>>> print(location2.raw)
{'place_id': '654513', 'osm_type': 'node', ...}
```

2.8 Librerie di data visualization

2.8.1 ReactMapGL

ReactMapGL è una libreria open source per React sviluppata da Uber che racchiude in componenti le funzionalità di Mapbox GL JS[64], una libreria JavaScript che permette la renderizzazione di mappe geografiche vettoriali interattive e personalizzabili utilizzando stili di mappa e dati vettoriali conformi alla specifica Mapbox e la libreria grafica WebGL.[65]

Il componente principale `InteractiveMap` è progettato per essere un componente senza stato. Il suo aspetto è completamente controllato dalle proprietà passate dal componente padre. In questa architettura, la transizione funziona come l'interazione: il componente notifica l'applicazione del-

l'intenzione di richiamare la funzione `onViewportChange()`, che gestisce il cambiamento del risultato grafico, ma spetta all'applicazione decidere come comportarsi.[66]

```
<ReactMapGL
    {...this.state.viewport}
    onViewportChange={(viewport) => this.setState({viewport})
   } />
```

È possibile aggiungere overlay di dati racchiudendo all'interno del componente principale dei componenti personalizzati. La libreria include alcuni overlay basilari, come `SVGOverlay`, e supporta i layer forniti dalla libreria deck.gl.

Gli overlay possono accedere all'area della mappa visualizzata correntemente attraverso l'oggetto Context di React.[67]

Tra gli altri componenti forniti dalla libreria ci sono `Marker` e `Popup`, usati per renderizzare in punti specifici della mappa rispettivamente icone e finestre popup con contenuto HTML personalizzato, e `NavigationControl`, un elemento interattivo fornito di pulsanti di zoom e bussola.[68]

2.8.2 D3.js

D3.js (**Data-Driven Documents**) è una libreria JavaScript che permette di collegare dati arbitrari a un DOM, per poi applicare al documento trasformazioni basate sui dati. Esempi sono la possibilità di genere una tabella HTML da un array numerico, o di usare gli stessi dati per creare un grafico a barre SVG interattivo con transizioni e interazioni fluide.[69]

Utilizza funzioni JavaScript prefatte per selezionare elementi del DOM, creare elementi SVG, aggiungere stili CSS, oppure transizioni, effetti di movimento e/o tooltip. In questo modo e così possibile generare ricche rappresentazioni grafiche di numeri, testi, mappe e diagrammi. I dati utilizzati possono essere in diversi formati, come JSON e CSV.

Il concetto centrale del design di D3 è permettere al programmatore di usare dei selettori, come per i CSS, per scegliere i nodi all'interno del DOM

e quindi usare operatori per manipolarli. Di seguito, viente riportato un esempio:

```
1 d3.selectAll("p")                // seleziona tutti gli
     elementi <p>
2    .style("color", "violet")      // imposta la chiave "
     color" con il valore "violet"
3    .attr("class", "squares")      // imposta l'attributo "
     class" con il valore "squares"
4    .attr("x", 50);                //  imposta l'attributo "x
     " con il valore 50px
```

Una volta che un set di dati viene associato a un documento, l'utilizzo di D3.js generalmente segue un modello secondo il quale per ogni elemento dell'insieme di dati associato vengono invocate una funzione esplicita `.enter()`, una funzione implicita "update" e una esplicita `.exit()`. Tutti i metodi concatenati dopo il comando `.enter()` verranno chiamati per ciascun elemento dell'insieme di dati per i quali non sia già presente un nodo DOM nella selezione (il `selectAll()` mostrato precedentemente). Allo stesso modo, la funzione di aggiornamento implicita viene chiamata su tutti i nodi selezionati esistenti per i quali esiste un elemento corrispondente nell'insieme di dati, e viene chiamata la `.exit()` su tutti i nodi selezionati esistenti che non abbiano un elemento nel set di dati a loro collegato.[70]

2.8.3 Recharts

Recharts è una libreria di grafici costruita su React e D3. Include un ampio numero di tipologie di grafici personalizzabili e fornisce diversi strumenti utili come griglie cartesiane, tooltip e contenitori responsive.[71]

Capitolo 3

Implementazione

Questo capitolo descrive i procedimenti effettuati per rappresentare in un'applicazione web i dati degli studenti a disposizione.

I dati mostrati nelle immagini presenti all'interno di questo capitolo non sono significativi dell'intero dataset.

3.1 Creazione e popolazione database

Le informazioni degli studenti rappresentate nel progetto sono state recuperate da un file CSV acquisito dall'Università di Bologna, contenente un campione minimo dei dati complessivi recuperati dalle risposte al questionario HousINg Unibo. Ciascun record del file contiene i seguenti campi:

- ciclo di laurea frequentato (Laurea, Laurea Magistrale, ecc.)
- corso di laurea
- anno di corso
- anno di nascita
- genere

- indirizzo e numero civico di abitazione a Bologna
- regione, provincia e comune di:
  - luogo di nascita,
  - residenza
  - domicilio
- sistemazione abitativa
- tasso di gradimento di permanenza a Bologna (molto, abbastanza, poco, per niente)
- token

La necessità di rappresentare geograficamente le posizioni dei luoghi di nascita, delle residenze e dei domicili ha portato alla loro geocodifica per ottenere le rispettive coordinate di latitudine e longitudine.

Queste informazioni sono poi state trasferite nel database "admin" di un server MongoDB installato localmente, interfacciabile utilizzando la stringa `mongodb://localhost:27017/admin` e avviato con l'eseguibile `mongod.exe`. È possibile eseguire questa operazione a partire da una richiesta di inizializzazione, eseguita all'accesso all'url `localhost:3001/api/init`. Viene utilizzata la libreria Mongoose per definire uno Schema per gli studenti, creare una connessione al database, e aggiornare quest'ultimo inserendo i dati facendo uso di Model appositi.

Di seguito, viene riportato lo Schema utilizzato per gli studenti:

```
new require("mongoose").Schema(
  {
    id: Number,
    cycle: String,
    degree: String,
    studyingYear: String,
    birthYear: Number,
    gender: String,
```

```
    birthplace: {
      region: String,
      province: String,
      municipality: String,
      latitude: Number,
      longitude: Number
    },
    residence: {
      region: String,
      province: String,
      municipality: String,
      latitude: Number,
      longitude: Number
    },
    domicile: {
      region: String,
      province: String,
      municipality: String,
      street: String,
      number: Number,
      latitude: Number,
      longitude: Number,
    },
    livingSituation: String,
    likingBologna: String,
    token: String
  }
);
```

In quanto le funzioni di interfaccia tra applicazione e database che sono a disposizione del prototipo Model richiedono parametri di tipo Object, è stato fatto uso di un file JSON ottenuto dalla conversione del file CSV iniziale, e che è stato generato utilizzando uno script Python appositamente creato. Come si può notare dallo Schema descritto precedentemente, le informazioni relative a luogo di nascita, residenza e domicilio sono state ulteriormente accorpate in oggetti interni per compattezza.

```
for row in csv_reader:
        if not row or not any(row):
            continue
        json_object = {}

        row_map = dict(zip(headers, map(str.strip, row)))

        for top in top_fields:
            if row_map[top] != "":
                if top is "birthYear":
                    json_object[top] = int(row_map[top])
                else:
                    json_object[top] = row_map[top]

        json_object["birthplace"] = {}
        if (row_map["birthplaceRegion"] != ""):
            json_object["birthplace"]["region"] = row_map["
    birthplaceRegion"]
        if (row_map["birthplaceProvince"] != ""):
            json_object["birthplace"]["province"] = row_map["
    birthplaceProvince"]
        if (row_map["birthplaceMunicipality"] != ""):
            json_object["birthplace"]["municipality"] =
    row_map["birthplaceMunicipality"]

        if json_object["birthplace"] == {}:
            del json_object["birthplace"]

     # qui operazioni analoghe per residenza e domicilio

        data.append(json_object)
```

Per ricavare latitudine e longitudine dei vari luoghi, è stato fatto uso di un altro script che, dato un file JSON in input, inserisce i parametri delle coordinate ottenute mediante le funzionalità della libreria GeoPy.

```
geolocator = Nominatim(user_agent="thesis-data-visualization
    ", timeout=5)
```

```
2
3 for s in data:
4     if "birthplace" in s:
5         if "latitude" not in s["birthplace"] or "longitude"
    not in s["birthplace"]:
6             if "municipality" in s["birthplace"]:
7                 addressString=s["birthplace"]["municipality"]
8                 try:
9                     location = geolocator.geocode(
    addressString)
10                    s["birthplace"]["latitude"] = location.
    latitude
11                    s["birthplace"]["longitude"] = location.
    longitude
12                except:
13                    pass
14
15   # qui operazioni analoghe per residenza e domicilio
```

Il database è stato popolato anche con dati relativi ai corsi di laurea offerti dall'Università di Bologna[72] e dati geometrici sui confini perimetrali dei quartieri di Bologna[73], attraverso metodi similari a quelli utilizzati per gli studenti.

3.2 Struttura dell'applicazione

L'applicazione ha un'architettura client-server: tutto ciò che riguarda l'interfaccia del sito è contenuto nella cartella `client`, mentre la connessione e le interazioni col database sono gestite all'interno della cartella `server`.

Sia nella cartella principale del progetto che nelle due sottocartelle è stato eseguito il comando `npm init`, che per ciascuna di loro ha creato il file `package.json`. Esso elenca le dipendenze del progetto, ovvero le librerie usate e le rispettive versioni minime richieste. Durante lo sviluppo, sono state man mano installate nuove librerie col comando `npm install <nome_package>`, che aggiorna il file package.json e le colloca nella cartella `node_modules`.

Il `package.json` nella cartella principale include tra le dipendenze unicamente la libreria `concurrently`, che si occupa di avviare sia il lato client che il lato server concorrentemente all'esecuzione di `npm start` (o di `npm run start-build` per buildare il frontend con i tool messi a disposizione da Create React App). I `package.json` nelle sottocartelle contengono informazioni sulle librerie utilizzate nel relativo contesto (client o server).

3.2.1 Server

Il lato server è concentrato nel file `index.js` nella cartella `server`, dove è stato utilizzato un oggetto Koa denominato `app` per contenere una serie di middleware eseguiti uno dietro l'altro all'effettuazione di una richiesta. Inizialmente vengono avviati middleware forniti da diversi moduli Koa: `cors`, utilizzato per rendere possibile un fetch dal client con origine (dominio, protocollo, porta) diversa, e `bodyParser`, per collocare il corpo dello stream di richiesta in `ctx.request.body`. A seguire, viene utilizzato un middleware chiamato `errorhandler` che ha il compito di restituire un risultato d'errore nel caso le operazioni dei middleware successivi non vadano a buon fine. Il corpo di questo middleware, così come quelli degli altri middleware originali, sono collocati nella cartella `middlewares`.

```
async (ctx, next) => {
    try {
      await next();
    } catch (err) {
      ctx.body = { success: false, error: err }
      ctx.status = 404;
    }
  }
```

Il middleware successivo, `db`, si occupa di creare una nuova connessione al database e renderla accessibile tramite `ctx.db`, per poi chiuderla al termine delle operazioni successive.

```
async (ctx, next) => {
    try {
```

```
        ctx.db = await mongoose.createConnection(DB_ROUTE, {
    useNewUrlParser: true, useFindAndModify: false });
        await next();
    } finally {
        if (ctx.db) {
            await ctx.db.close();
        }
    }
};
```

Vengono poi configurate due route relative rispettivamente all'inizializzazione dei dati nel database (`/api/init`) e al loro recupero (`/api/get`), ed importate da `middlewares/routes` . La prima definisce un modello per ciascuna collection nel database (Student, Neighborhood, DegreesSeat; studenti, quartieri, corsi di laurea) utilizzando i relativi Schema nella cartella `schemas` , e li utilizza per resettare i dati attualmente memorizzati e caricare quelli contenuti nei relativi file JSON collocati nella cartella `collections` .

```
// x = student/neighborhood/degreesSeat

async (ctx, next) => {
        var xsModel = await ctx.db.model("X", require("../../
    schemas/x"))
        var xsCollection = require("../../collections/xs.json
    ");

        await xsModel.collection.deleteMany({});

        for (const document of xsCollection) {
            await (new xsModel(document)).save();
        };

        ctx.body = { success: true };
        ctx.status = 200;

        await next();
}
```

L'altra route si occupa semplicemente di recuperare i dati nel database e restituirli nel corpo dell'oggetto risposta.

```
async (ctx, next) => {
      var studentsModel = await ctx.db.model("Student",
   require("../../schemas/student"))
      var neighborhoodsModel = await ctx.db.model("
   Neighborhood", require('../../schemas/neighborhood"))
      var degreesSeatsModel = await ctx.db.model("DegreesSeat
   ", require("../../schemas/degreesSeat"))
      var students = await studentsModel.find().exec();
      var neighborhoods = await neighborhoodsModel.find().
   exec();
      var degreesSeats = await degreesSeatsModel.find().exec
   ();

      ctx.body = {
            success: true,
            students: students,
            neighborhoods: neighborhoods,
            degreesSeats: degreesSeats
      };
      ctx.status = 200;
}
```

Tutte funzioni di connessione e interazione col database, come `find()` e `deleteMany()`, fanno parte della libreria Mongoose.

L'ultima riga del file `server/index.js` si occupa di mettere in ascolto il server alla porta 3001 mediante il metodo `app.listen()`.

3.2.2 Client

Il lato client è stato costruito con l'ambiente Create React App, una libreria ideata appositamente per fornire una serie di tool selezionati e già configurati per creare applicazioni con React. All'avvio, viene renderizzato un componente `Router` contenente una `Route` per il componente `App` principale al percorso `/` all'interno dell'elemento con id `root` del file `index.html`.

```
ReactDOM.render(<Router>
                    <Route exact path="/" component={() => <App
    />} />
                  </Router>, document.getElementById("root"));
serviceWorker();
```

Il componente `App` renderizza tre sottocomponenti: `Navbar`, una barra di navigazione che fa uso di componenti `Link` per reindirizzare la pagina alle sue sottosezioni con ancoraggi; `MapPanel`, ovvero la sottointerfaccia di visualizzazione dei dati degli studenti in una mappa geografica; `ChartPanel`, che racchiude la rappresentazioni dei dati sotto forma di grafici. Essi sono racchiusi in `React.Fragment`, un componente utilizzato per racchiudere molteplici elementi senza generare nodi aggiuntivi nel DOM, come ad esempio dei `div`.

Una volta che `App` è stato montato (quando gli elementi HTML relativi sono stati aggiunti al DOM), viene effettuato un fetch al server per ricavare tutti i dati presenti nel database relativi agli studenti, ai quartieri e ai corsi di laurea, per poi passarli ai componenti figli `MapPanel` e `ChartPanel` utilizzando delle funzioni a loro disposizione richiamabili attraverso delle refs, proprietà di React create e poi assegnate ai componenti figli. Oltre a questi dati, vengono passate le `props` prese in ingresso da `App` relative alle opzioni selezionate inizialmente dai `ReactResponsiveSelect`.

La gestione di props mancanti e di tipo errato avviene attraverso, rispettivamente, le proprietà statiche dei componenti React `defaultProps` e `propTypes`.

I componenti dell'applicazione sono contenuti nella sottocartella `components`. Sono presenti anche la cartella `styles`, che racchiude i file `.module.css` e `.module.scss` con gli stili CSS utilizzati da alcuni componenti, e la cartella `utilities`, contenente funzioni di utilità richiamate più volte.

Oltre agli stili CSS menzionati precendentemente, il progetto fa uso del framework Bootstrap, installato come le altre librerie con `npm`.

3.3 Sviluppo della mappa

Per rappresentare i dati geografici degli studenti si è pensato di mettere a disposizione dell'utente una mappa con metodi di visualizzazione e categorie di dati intercambiabili, selezionabili da appositi componenti select:

- modalità di visualizzazione: controlla la tipologia di rappresentazione dei dati utilizzata, la quale può essere:
  - marker: ciascuna posizione è rappresentata da un'icona circolare; se le posizioni sono troppo vicine tra loro, relativamente al grado di zoom corrente della mappa, le icone delle singole posizioni verranno sostituite da un'unica icona di maggiore circonferenza e riportante il numero delle posizioni che comprende (cluster).
  - heatmap: le posizioni vengono rappresentate sotto forma di mappa di calore; i colori saranno più intensi nelle aree con un numero di posizioni maggiore.
  - quartieri di Bologna: viene rappresentata una mappa coropletica nell'area relativa ai quartieri di Bologna; l'intensità del colore del quartiere è proporzionale alla quantità delle posizioni al suo interno rispetto al totale.
- tipo di posizione: controlla quale categoria di dati geografici deve essere rappresentata (luogo di nascita, residenza o domicilio).
- parametro di visualizzazione: selezionabile solo in modalità marker, assegna un colore diverso alle icone in base al valore della proprietà selezionata dello studente (distanza dalla sede di corso, corso di laurea, anno di corso e altre categorie di dati analoghe a quelle memorizzate nel database); nel caso un cluster comprenda posizioni relative a studenti con valori diversi, avrà il colore di default.
- secondo parametro di visualizzazione: permette di assegnare un colore diverso alle icone per ciascuna combinazione esistente di valori della

seconda proprietà selezionata con valori della prima proprietà; appare solo a primo parametro di visualizzazione selezionato e con proprietà diversa da "distanza dalla sede di corso".

Il componente che racchiude tutto questo è `MapPanel`: comprende il componente `WorldMap`, quattro `ReactResponsiveSelect` e `MapLegend`. Per suddividere lo spazio dei componenti, sono stati usati utilizzati dei sistemi a griglia di Bootstrap: nei viewport con larghezza maggiore di 576 pixel (`sm`) la mappa occupa 3/4 dell'elemento contenitore e il restante 1/4 è occupato dai tre select e dalla legenda, mentre in viewport con larghezza minore o uguale a 576px le due parti occupano l'intera larghezza con la mappa sopra il resto. In quest'ultimo caso, inoltre, il componente `MapLegend` appare come una finesta modale visualizzabile cliccando su un apposita checkbox.

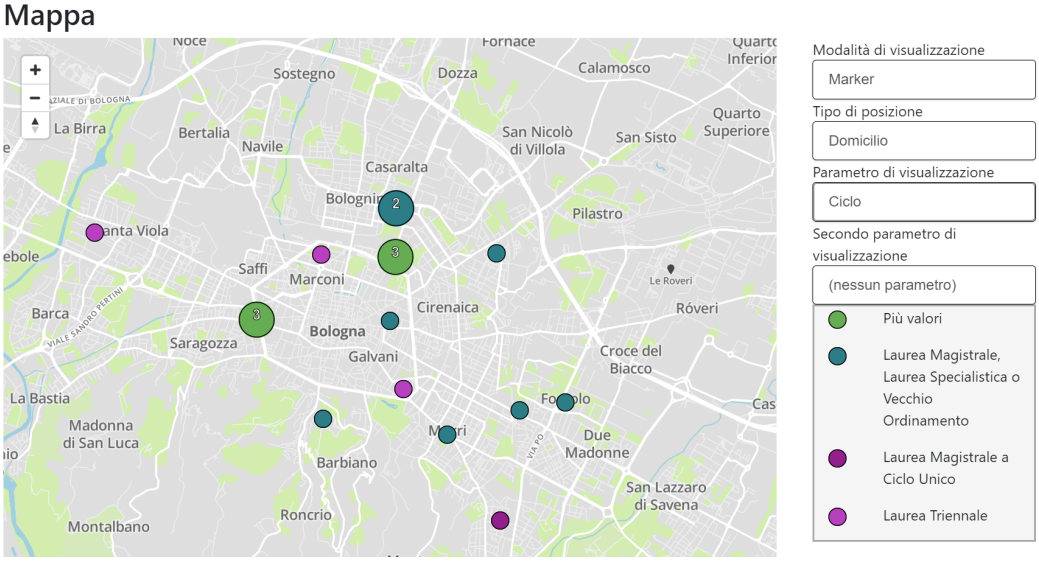

Figura 3.1: Il risultato della renderizzazione del componente MapPanel.

`MapPanel` prende in input tramite props le informazioni degli studenti, dei quartieri, dei corsi di laurea e dei parametri dei select da rendere selezionati al termine del montaggio dei componenti, e le assegna allo stato per gestire le loro eventuali modifiche. La modifica dello stato può avvenire sia trami-

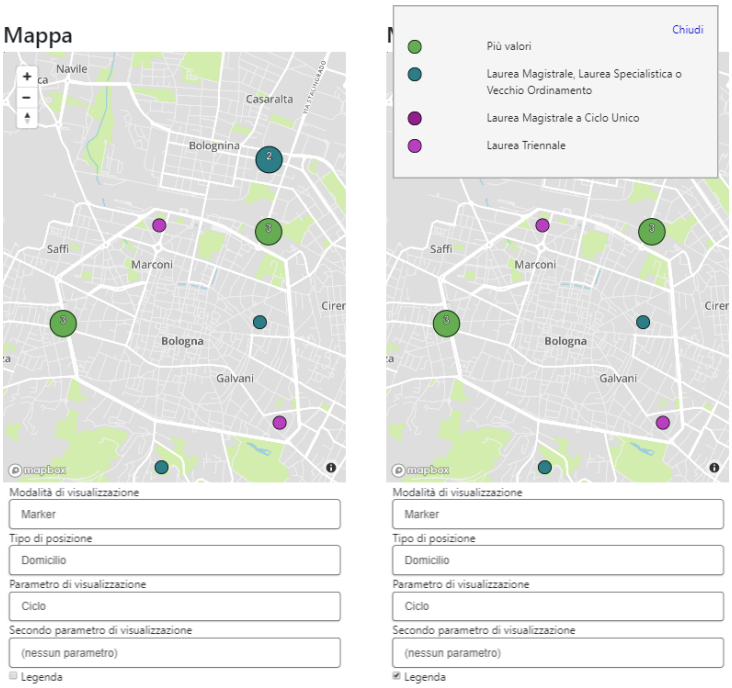

Figura 3.2: Il risultato della renderizzazione del componente MapPanel su risoluzione mobile, rispettivamente con legenda abilitata e disabilitata.

te la chiamata della funzione `updateWorldMapState()` da parte di `App`, che a seguito della selezione in uno dei quattro select a disposizione. Congiuntamente, verranno richiamate tramite refs le funzioni `updateState()` dei componenti figli `WorldMap` e `MapLegend` atte ad aggiornare lo stato di questi ultimi con le nuove informazioni.

`WorldMap` restituisce un componente `ReactMapGL` che, oltre ad avere come proprietà informazioni sulla viewport, sull'accesso e al caricamento della mappa, ha tra i figli un `NavigationControl` fornito dalla libreria e un componente dipendente dalla modalità di visualizzazione selezionata e quindi memorizzata nello stato.

In modalità marker (`this.state.mode === "markers"`), verrà renderizzato un componente `Cluster` [74][75] che racchiude e renderizza un componente `Marker` per ciascuno studente rappresentabile, e che fa uso della

libreria Supercluster per accorpare in un unico `Marker` con proprietà diverse le posizioni vicine tra loro in base alle proprietà impostate (livelli di zoom minimo e massimo a cui viene generato il cluster, raggio di azione, ecc.).

Ciascun componente `Marker` ha come proprietà le coordinate della posizione a esso associata, e ha come figlio un componente Pin, che consiste in un'icona circolare a cui viene assegnato un colore e una funzione attivata dopo un click su di essa.

Generalmente il colore di un `Pin` è calcolato utilizzando come riferimento le stringhe dei valori dei parametri di visualizzazioni selezionati. Nel caso il parametro selezionato sia la distanza dalla sede di corso, viene utilizzata la funzione `scaleLinear()` offerta dalla libreria D3 per regolare l'intensità del colore in base al valore dell'elemento, calcolato utilizzando una funzione che restituisce la distanza geografica in chilometri tra la posizione del marker e quella della sede del corso che lo studente associato frequenta (memorizzata in `this.state.degreesSeats`).

Il colore dei `Pin` associati ai cluster è calcolato in modo analogo utilizzando come riferimento il primo degli studenti in esso compresi, ma solamente se tutti i valori del parametro degli studenti sono uguali; in caso contrario, verrà utilizzato il colore di default. Per colore di default si intende il colore generato utilizzando come parametro la stringa `"all"`.

Sia `Cluster` che `Marker` e `Pin` estendono la classe `React.PureComponent`: a differenza dei componenti normali, implementano il metodo `shouldComponentUpdate()` con una comparazione "shallow" delle props e dello state, ovvero confrontando gli oggetti correnti con quelli nuovi solamente tramite chiavi univoche e non nel loro contenuto. Quando si ha la certezza che il risultato della renderizzazione di un componente sia sempre lo stesso a partire dagli stessi valori di props e state, è conveniente utilizzare questa classe per avere un miglioramento delle performance in alcuni casi.[76]

Nella funzione di render, viene renderizzato anche il contenuto di `this.state.popup`, inizialmente `null` e modificato al click su un componente `Marker` assegnan-

do allo stato un componente `Popup`.

```
handleMarkerClick(student, popupStrings) {
    let popupParametersCount = popupStrings.filter(popup => popup.parameterString)
                                           .map(popup => popup.parameterString)
                                           .reduce((a, c) => {a[c] = (a[c] || 0) + 1; return a}, {});

    if (student && student[this.state.type] && popupStrings.length > 0) {
      this.setState({
        popup: <Popup className={styles.popup + " " + styles.fadeIn}
                tipSize={5}
                anchor="bottom-right"
                longitude={student[this.state.type].longitude}
                latitude={student[this.state.type].latitude}
                onClose={() => this.setState({popup: null})}
                closeOnClick={true}>
                  <React.Fragment>
                  <div className={styles.address}>{popupStrings[0].addressString}</div>
                  <div className={styles.parameterList}>
                    {Object.keys(popupParametersCount).map((pp, key) =>
                      <div key={"parameter-"+key} className={styles.parameter}>
                        {pp + ((popupParametersCount[pp] > 1) ? (" (" + popupParametersCount[pp] + ")") : "")}
                      </div>
                    )}
                  </div>
                </React.Fragment>
              </Popup>
      });
```

```
    }
  }
```

Per `popupStrings` si intende un array di oggetti col formato

```
{
  addressString: String,
  parameterString: String
}
```

ricavato con la seguente funzione:

```
getPopupStrings(students, degreesSeats, type, parameter,
    parameter2) {
    let popupStrings = [];
    students.forEach(student => {
      let addressString = "";
      let parameterString = "";
      if (student[type]) {
        if (type === "domicile" && student[type].street) {
          addressString += (student[type].street + " ")
        }
        if (type === "domicile" && student[type].number) {
          addressString += (student[type].number + ", ")
        }
        if (student[type].municipality) {
          addressString += student[type].municipality
        }
      }
      if (parameter === "seatDistance" && student[type] &&
    student[type].latitude && student[type].longitude) {
        parameterString += (getSeatDistance(student, type,
    degreesSeats)+"km")
      } else if (parameter && parameter !== "none" && student
    [parameter]) {
        parameterString += student[parameter]
      }
      if (parameter2 === "seatDistance" && student[type] &&
    student[type].latitude && student[type].longitude) {
```

```
23          parameterString += (" - " + getSeatDistance(student,
    type, degreesSeats)+"km")
24        } else if (parameter2 && parameter2 !== "none" &&
    student[parameter2]) {
25          parameterString += (" - " + student[parameter2])
26        }
27        popupStrings.push({
28          addressString: addressString,
29          parameterString: parameterString
30        })
31      })
32      return popupStrings;
33  }
```

In quanto è possibile che esistano cluster relativi a più posizioni con le stesse identiche coordinate (ad esempio la stessa città), si è ritenuto necessario gestire la comparsa di un popup anche per i cluster che si trovano in questa situazione: viene riutilizzato il codice della funzione `handleMarkerClick()`, usando come riferimento per le coordinate la prima posizione nell'array delle posizioni degli studenti. Indipendentemente da ciò, un click su un cluster porta a uno zoom della mappa per visualizzare i marker delle posizioni singole, se possibile.

```
1  handleClusterClick(clusterProps) {
2      if (clusterProps.points.length === 1) {
3        let clusterStudents = this.getClusterStudents(
    clusterProps);
4        let popupStrings = this.getPopupStrings(clusterStudents
    , this.state.degreesSeats, this.state.type, this.state.
    parameter, this.state.parameter2)
5        if (clusterStudents.length > 0 && popupStrings.length >
     0) {
6          this.handleMarkerClick(clusterStudents[0],
    popupStrings)
7        }
8      }
9
```

```
10      if (this.state.viewport.zoom < this.state.viewport.
    maxZoom) {
11        this.setState(prevState => ({ viewport: {
12          ...prevState.viewport,
13          longitude: clusterProps.cluster.geometry.coordinates
    [0],
14          latitude: clusterProps.cluster.geometry.coordinates
    [1],
15          zoom: (prevState.viewport.zoom + 2 < prevState.
    viewport.maxZoom) ? (prevState.viewport.zoom + 2) : (this.
    state.viewport.maxZoom)
16        }}))
17      }
18  }
```

`clusterProps.points` è un array delle posizioni distinte rappresentate nel cluster: ciò vuol dire che se un cluster rappresenta 30 posizioni con le stesse identiche coordinate, `clusterProps.points` avrà una lunghezza pari a 1.

Nel caso la modalità selezionata sia heatmap (`this.state.mode === "heatmap"`), verrà semplicemente renderizzato un componente `HeatmapOverlay` della libreria `react-map-gl-heatmap-overlay` che prende in ingresso le coordinate delle posizioni del tipo selezionato e le informazioni di viewport.

Per quanto riguarda la modalità relativa ai quartieri di Bologna (`this.state.mode === "choropleth"`), verrà renderizzato un componente `ChoroplethOverlay` , che prende in ingresso l'insieme degli oggetti GeoJSON Feature relativi ai quartieri memorizzati in `this.state.neighborhoods` per rappresentare le rispettive figure poligonali. Ciascun poligono è riempito da un colore con intensità dipendente dall'attributo `properties.value` della Feature, assegnato durante l'aggiornamento dello stato e che rappresenta il numero di posizioni comprese nell'area del quartiere:

```
1  this.setState({
2        ...
```

```
3       neighborhoods: (neighborhoods) ? neighborhoods.map((
    feature) => {
4           const f = feature;
5           f.properties.value = (type)
6             ? (students.filter((student) =>
7               (student[type] &&
8               student[type].longitude && student[type].
    latitude &&
9               geoContains(feature, [student[type].longitude,
    student[type].latitude]))).length)
10            : 0;
11          return f;
12      }) : neighborhoods,
13  ...
14 })
```

`geoContains` è una funzione fornita dalla libreria D3 che restituisce un valore booleano `true` se la feature passata come primo parametro comprende il punto con le coordinate passate nel secondo parametro, altrimenti `false`.

`MapLegend` è un componente che mostra una legenda dei colori utilizzati nella mappa, visibile in modalità "Marker" e "Quartieri di Bologna". Viene utilizzato un sistema a griglia Bootstrap, con una colonna per i `Pin` rappresentanti i colori e una per il rispettivo valore.

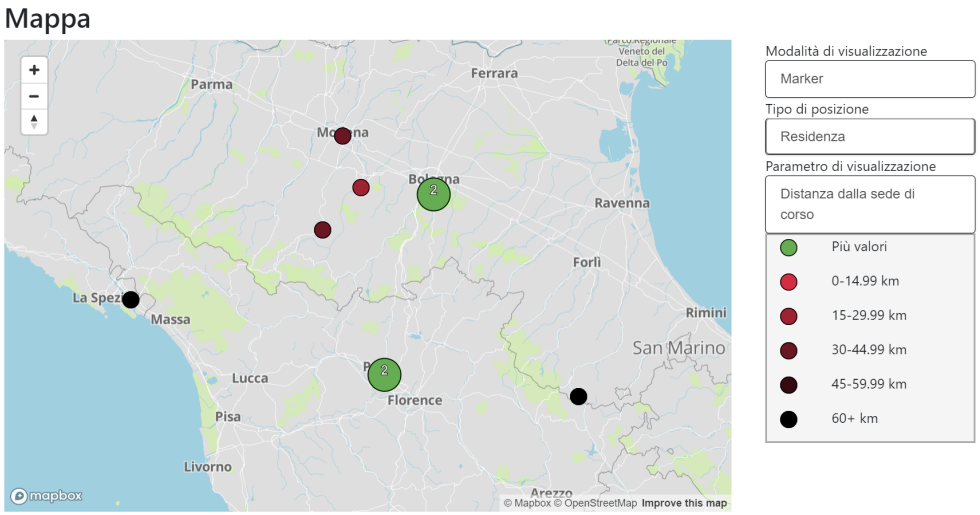

Figura 3.3: Il risultato della renderizzazione del componente MapPanel selezionando il parametro relativo alla distanza dalla sede di corso.

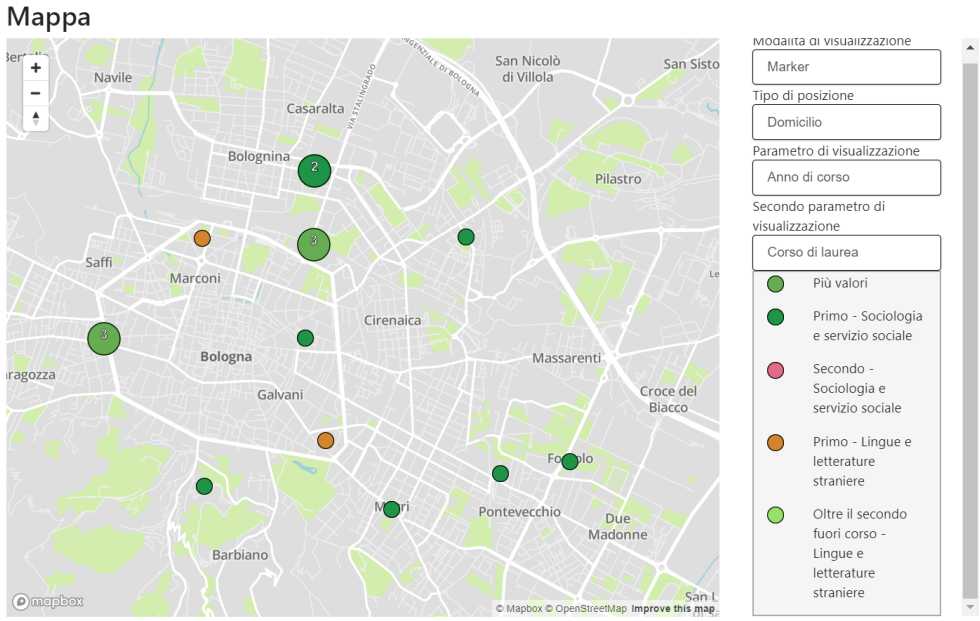

Figura 3.4: Un possibile risultato della renderizzazione del componente MapPanel selezionando due parametri di visualizzazione.

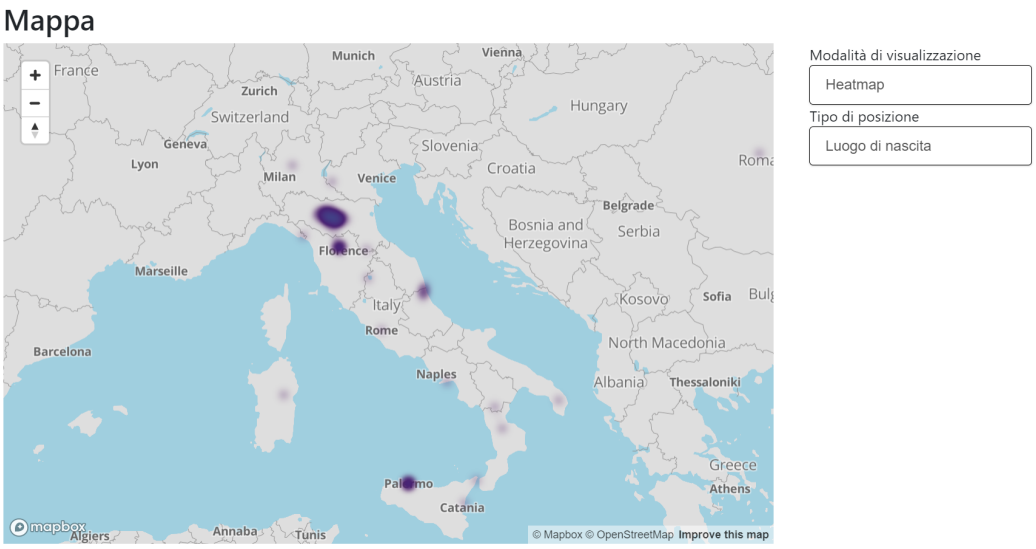

Figura 3.5: Il risultato della renderizzazione del componente MapPanel in modalità heatmap.

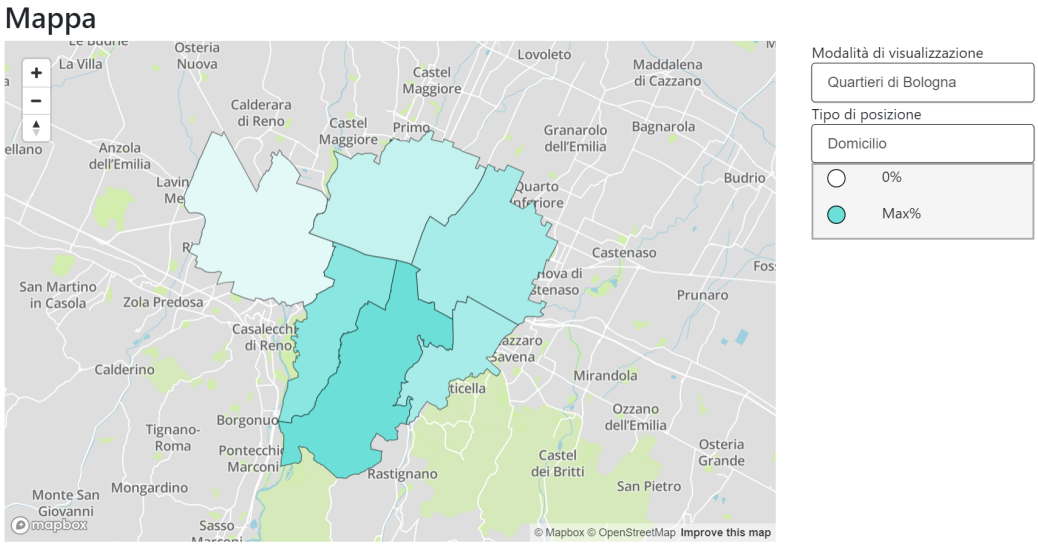

Figura 3.6: Il risultato della renderizzazione del componente MapPanel in modalità quartieri di Bologna.

3.4 Sviluppo dei grafici

Per confrontare la quantità dei dati degli studenti rispetto a determinati parametri, è stata sviluppata un'interfaccia di data visualization utilizzando grafici adatti per la tipologia di dati a disposizione. Si è optato per utilizzare unicamente grafici a barre e grafici a torta in quanto, come analizzato nella sezione 1.2.2, altre tipologie di grafici come istogrammi e grafici a linee non risultano idonei alla visualizzazione di dati discreti e non temporali.

I parametri rispetto ai quali è possibile suddividere i dati degli studenti corrispondono perlopiù agli attributi degli studenti memorizzati nel database (corso di laurea, anno di corso, ecc.); a questi si aggiungono le divisioni per quartieri e per distanza dalla sede di corso dei tipi di posizione (luogo di nascita, residenza, domicilio).

Il componente relativo a questa sezione è `ChartPanel`, che comprende il componente `Chart` e due `ReactResponsiveSelect`, di cui uno controlla la tipologia del grafico visualizzato e l'altro il parametro di visualizzazione. La loro disposizione è similare a quella dei componenti figli di `MapPanel`, in quanto è stato applicato nuovamente il sistema a griglia di Bootstrap: se la viewport ha una larghezza maggiore di 576 pixel (`sm`) `Chart` occupa 3/4 dell'elemento contenitore , altrimenti l'intero spazio orizzontale con i select sottostanti.

Le proprietà di ciascuna singola barra o fetta possono essere visualizzate spostando il cursore del mouse su di essa (o, se su mobile, cliccandoci sopra), tramite un tooltip.

`ChartPanel` prende in input tramite props le informazioni degli studenti, dei quartieri, dei corsi di laurea e dei parametri dei select da rendere selezionati all'inizio, e le assegna allo stato per gestire le loro eventuali modifiche. La modifica dello stato può avvenire sia tramite la chiamata della funzione `updateChartState()` da parte di `App`, che attraverso la selezione in uno dei due select a disposizione. In entrambi i casi, verrà richiamata tramite ref la funzione `updateState()` del componente figlio `Chart`, che aggiornerà il suo stato con il tipo del grafico e un array di oggetti contenenti le infor-

mazioni da rappresentare. Questi oggetti sono ricavati tramite una funzione apposita e hanno il seguente formato:

```
{
  name: element,
  n: count
}
```

Gli attributi `name` corrispondono ai valori relativi al parametro selezionato esistenti, e `n` alla rispettiva quantità.

Il componente `Chart` fa uso di componenti ricavati dalla libreria Recharts per rappresentare grafici a torta o a barre in base al valore contenuto nel suo stato; l'array di dati descritto precedentemente può essere direttamente passato come proprietà `data` di questi componenti senza la necessità di applicare ulteriori trasformazioni.

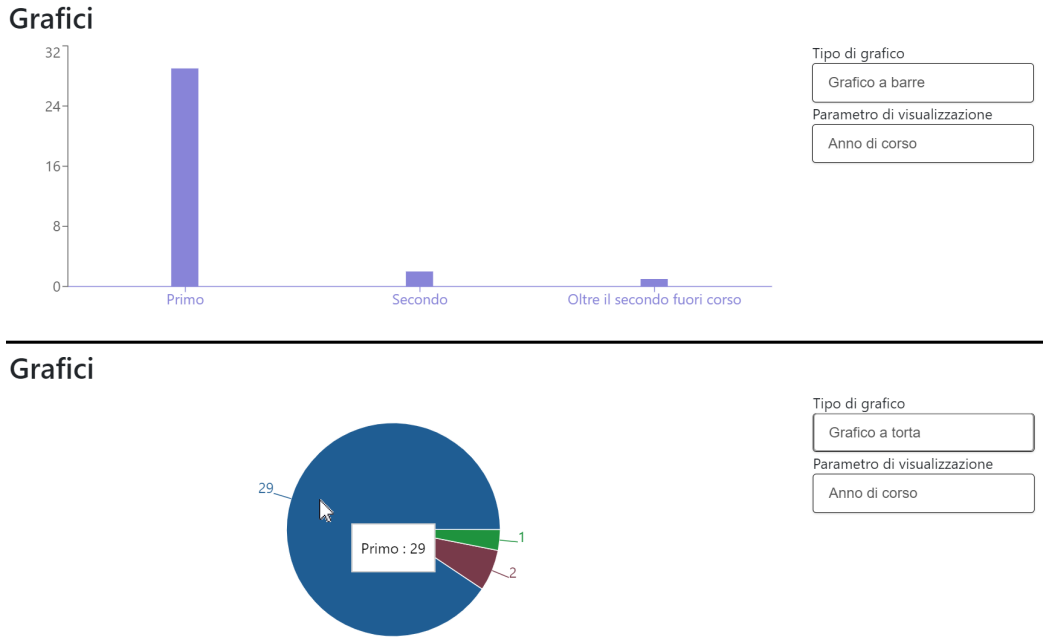

Figura 3.7: I risultati delle renderizzazioni del componente ChartPanel rappresentanti proprietà degli studenti.

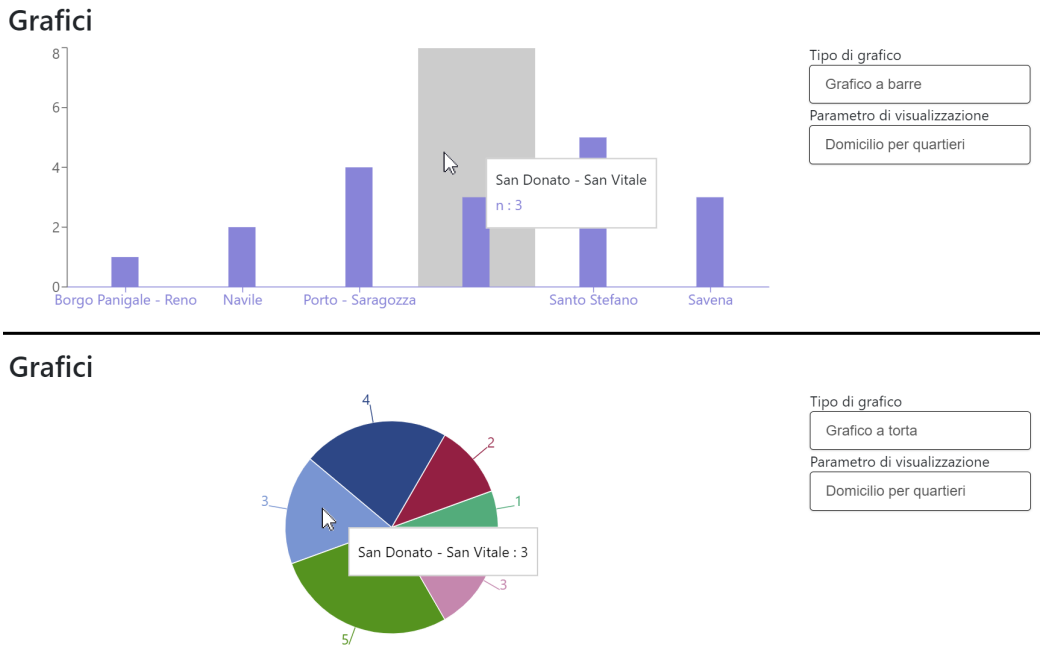

Figura 3.8: I risultati delle renderizzazioni del componente ChartPanel rappresentanti le posizioni relative agli studenti filtrate per quartieri.

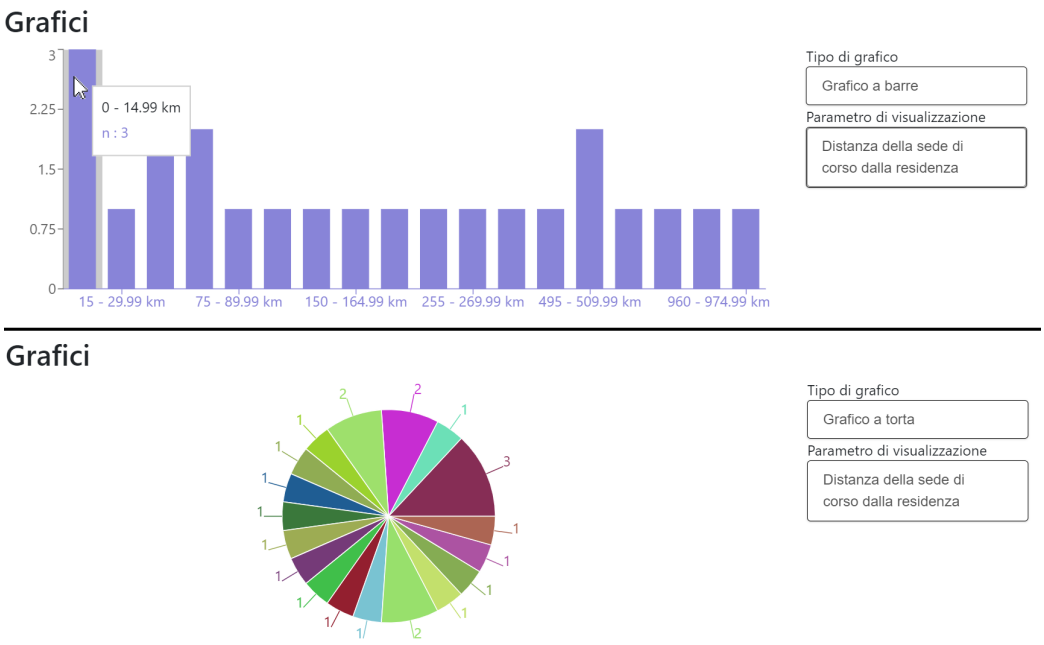

Figura 3.9: I risultati delle renderizzazioni del componente ChartPanel rappresentanti le distanze delle sedi di corso dalle posizioni relative agli studenti.

Conclusioni

L'obiettivo di questa tesi è quello di illustrare le tematiche e le fasi di lavoro che hanno interessato lo sviluppo di un'applicazione web finalizzata a rappresentare i dati raccolti dall'indagine HousINg Unibo, ponendo particolare rilievo alle proprietà geografiche geocodificate. Per implementare in modo adeguato rappresentazioni visuali e interattive, è stata posta un'attenzione particolare alle caratteristiche delle diverse tipologie di grafici, in modo tale da scegliere per lo sviluppo quelle che avrebbero rappresentato al meglio i dati presi in considerazione.

Lo sviluppo dell'elaborato di tesi ha fatto uso, come occasione di crescita personale dell'autore, di diverse tecnologie inizialmente poco conosciute e non studiate nell'ambito dei corsi universitari seguiti, in particolare: MongoDB, React, Node.js e Koa.js, Python.

L'applicazione sviluppata è stata testata su più browser (Google Chrome, Mozilla Firefox, Microsoft Edge), risulta funzionante e restituisce risultati corretti se configurata adeguatamente. Tuttavia, utilizzando quantità di dati elevate può verificarsi un fenomeno di lag durante lo scorrimento della mappa geografica, a seguito della renderizzazione degli overlay di dati a ogni spostamento da parte dei componenti forniti dalla libreria ReactMapGL. Avendo lavorato su una quantità di dati ridotta, questo problema è stato scoperto a fase avanzata di sviluppo e in futuro sarebbe interessante rielaborare il codice per risolverlo.

Bibliografia

[1] What is spatial data? https://www.safe.com/what-is/spatial-data/.

[2] Geocoding: How geocoding works. https://guides.library.illinois.edu/c.php?g=741567&p=5331379.

[3] Gerard Rushton, Marc P. Armstrong, Josephine Gittler, Barry R. Greene, Claire E. Pavlik, Michele M. West, and Dale L. Zimmerman. *Geocoding Health Data*, chapter 1.3.4, Issues with the spatial statistical analysis of geocoded data. CRC Press, novembre 2007.

[4] Sabina Ventura. *Linee guida per il geocoding e la georeferenziazione dei beni in SIGECweb.* Istituto Centrale per il Catalogo e la Documentazione, 1.1 edition, marzo 2014.

[5] Georeferencing. https://www.ordnancesurvey.co.uk/support/understanding-gis/georeferencing.html.

[6] Muzammil Khan and Sarwar Shah Khan. Data and information visualization methods, and interactive mechanisms: A survey. *International Journal of Computer Applications (0975 – 8887)*, novembre 2011.

[7] Luca Chittaro. Visualizing information on mobile devices. *Computer, Vol. 39 Iss. 3, pp. 3 - 4*, marzo 2006.

[8] Valinda Chan. Getting it right: why infographics are not the same as data visualizations. https://blog.prototypr.io/

getting-it-right-why-infographics-are-not-the-same-as-data-visualizations-a23 giugno 2017.

[9] Mico Yuk and Stephanie Diamond. Understanding the difference between data visualization and infographics. https://www.dummies.com/programming/big-data/big-data-visualization/understanding-the-difference-between-data-visualization-and-infographics/.

[10] Andy Kirk. *Data Visualization: A Handbook for Data Driven Design*, chapter Defining Data Visualization, The components of understanding. SAGE Publications, giugno 2016.

[11] Manuela Aparicio and Carlos J. Costa. Data visualization. *Communication Design Quarterly*, novembre 2014.

[12] Data visualization: What it is and why it matters. https://www.sas.com/en_us/insights/big-data/data-visualization.html.

[13] Steve Jones. How to overcome data visualisation problems. https://www.smartdatacollective.com/overcome-data-visualisation-problems/, luglio 2017.

[14] Grafico a barre. https://help.qlik.com/it-IT/sense/June2019/Subsystems/Hub/Content/Sense_Hub/Visualizations/Bar-Chart/bar-chart.htm.

[15] Robert Grant. *Data Visualization: Charts, Maps and Interactive Graphics*, chapter Maps and networks. CRC Press, novembre 2018.

[16] 5 popular thematic map types and techniques for spatial data. https://carto.com/blog/popular-thematic-map-types-techniques-spatial-data/.

[17] Dot distribution vs graduated symbols vs proportional symbol maps. https://gisgeography.com/dot-distribution-graduated-symbols-proportional-symbol-maps/.

[18] Vladimir Fedak. Big data: Information visualization techniques. https://towardsdatascience.com/big-data-information-visualization-techniques-f29150dea190, gennaio 2018.

[19] Heatmap on a map in python. https://datascience.stackexchange.com/questions/14774/heatmap-on-a-map-in-python.

[20] Andrea De Mauro, Marco Greco, and Andrea Grimaldi. A formal definition of big data based on its essential features. *Library Review, Vol. 65 Iss: 3, pp.122 – 135*, marzo 2016.

[21] The 3vs that define big data. https://www.datasciencecentral.com/forum/topics/the-3vs-that-define-big-data.

[22] Bernard Marr. Why only one of the 5 vs of big data really matters. https://www.ibmbigdatahub.com/blog/why-only-one-5-vs-big-data-really-matters, marzo 2015.

[23] Rob Kitchin. *The Data Revolution: Big Data, Open Data, Data Infrastructures and Their Consequences*, chapter Big Data. SAGE Publications, agosto 2014.

[24] The open definition. http://opendefinition.org/.

[25] Cosa sono i dati aperti (open data)? http://opendatahandbook.org/guide/it/what-is-open-data/.

[26] Come aprire i dati. http://opendatahandbook.org/guide/en/how-to-open-up-data/.

[27] Ernesto Belisario. Licenze per il riutilizzo dei dati pubblici e open data. http://formazione.formez.it/sites/all/files/open_data_-_licenze_per_il_riutilizzo_dei_dati_pubblici_e_open_data.pdf, novembre 2014.

[28] Open data: ecco il portale dei dati aperti dell'università di bologna. https://magazine.unibo.it/archivio/2017/12/04/open-data-ecco-il-portale-dei-dati-aperti-unibo.

[29] What is html? https://www.yourhtmlsource.com/starthere/whatishtml.html.

[30] Html. https://it.wikipedia.org/wiki/HTML#Descrizione.

[31] What is a web browser? http://www.corelangs.com/html/introduction/web-browsers.html.

[32] Cody Arsenault. Html vs html5 - what's the difference? https://www.keycdn.com/blog/html-vs-html5, gennaio 2017.

[33] Scott Morris. Tech 101: The ultimate guide to css. https://skillcrush.com/2012/04/03/css/, novembre 2018.

[34] Css selectors. https://developer.mozilla.org/it/docs/Web/CSS/CSS_Selectors#Basic_Selectors.

[35] David Sawyer McFarland. *CSS: The Missing Manual*, chapter Anatomy of a Style. O'Reilly Media, 4th edition, agosto 2015.

[36] Types of css (cascading style sheet). https://www.geeksforgeeks.org/types-of-css-cascading-style-sheet/.

[37] Robin Rendle. What are css modules? https://css-tricks.com/css-modules-part-1-need/, aprile 2016.

[38] Andrew Farmer. What are css modules? a visual introduction. https://www.javascriptstuff.com/what-are-css-modules/, aprile 2016.

[39] Massimo Carnevale. https://www.ueppy.com/news/fogli-di-stile/introduzione-a-sass-e-less-come-utilizzare-al-meglio-i-css.html. https://www.ueppy.com/news/fogli-di-stile/

introduzione-a-sass-e-less-come-utilizzare-al-meglio-i-css.html, settembre 2015.

[40] What's the difference between scss and sass? https://stackoverflow.com/questions/5654447/whats-the-difference-between-scss-and-sass.

[41] About bootstrap. https://stackoverflow.com/tags/twitter-bootstrap/info.

[42] About javascript. https://stackoverflow.com/tags/javascript/info.

[43] React. https://it.reactjs.org/.

[44] Nitin Pandit. What is reactjs and why should we use it? https://www.c-sharpcorner.com/article/what-and-why-reactjs/, maggio 2017.

[45] About node.js. https://stackoverflow.com/tags/node.js/info.

[46] Informazioni su node.js. https://nodejs.org/it/about/.

[47] The node.js event loop, timers, and process.nexttick(). https://nodejs.org/it/docs/guides/event-loop-timers-and-nexttick/.

[48] Claudio Marotta. Node.js event loop: cos'è e come funziona. https://www.mrwebmaster.it/javascript/node-js-event-loop-come-funziona_12434.html, dicembre 2017.

[49] Koa. https://koajs.com/.

[50] Selvaganesh. How node js middleware works? https://medium.com/@selvaganesh93/how-node-js-middleware-works-d8e02a936113, giugno 2018.

[51] Dave Swersky. Koa vs express in nodejs: 2018 edition. https://raygun.com/blog/koa-vs-express/, maggio 2018.

[52] Promise. https://developer.mozilla.org/it/docs/Web/JavaScript/Reference/Global_Objects/Promise.

[53] Async/await. https://javascript.info/async-await.

[54] Csv. https://it.wikipedia.org/wiki/Comma-separated_values.

[55] Introduzione a json. https://www.json.org/json-it.html.

[56] Geojson. https://geojson.org/.

[57] Margaret Rouse. Mongodb. https://searchdatamanagement.techtarget.com/definition/MongoDB.

[58] Nick Karnik. Introduction to mongoose for mongodb. https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/, febbraio 2018.

[59] Simon Holmes. *Mongoose for Application Development*, chapter The cornerstones of Mongoose. Packt Publishing, agosto 2013.

[60] Connection string uri format. https://docs.mongodb.com/manual/reference/connection-string/.

[61] Simon Holmes. *Mongoose for Application Development*, chapter Establishing a Database Connection. Packt Publishing, agosto 2013.

[62] About python. https://stackoverflow.com/tags/python/info.

[63] Welcome to geopy's documentation! https://geopy.readthedocs.io/en/stable/.

[64] Introduction. https://uber.github.io/react-map-gl/#/Documentation/introduction/introduction.

[65] Mapbox gl js readme. https://github.com/mapbox/mapbox-gl-js/blob/master/README.md.

[66] State management. https://uber.github.io/react-map-gl/#/Documentation/advanced/custom-state-management.

[67] Custom overlays. https://uber.github.io/react-map-gl/#/Documentation/advanced/custom-overlays.

[68] Custom components. https://uber.github.io/react-map-gl/#/Documentation/advanced/custom-components.

[69] Data-driven documents. https://d3js.org/.

[70] D3. https://it.wikipedia.org/wiki/D3.js.

[71] Recharts. http://recharts.org/en-US/.

[72] Corsi di studio 2019/2020. https://dati.unibo.it/dataset/degree-programmes/resource/corsi_2019_it.

[73] Quartieri di bologna. http://dati.comune.bologna.it/node/1183.

[74] uber/react-map-gl cluster. https://github.com/uber/react-map-gl/issues/507#issuecomment-424860068.

[75] urbica/react-map-gl-cluster cluster. https://github.com/urbica/react-map-gl-cluster/blob/master/src/components/Cluster/index.js.

[76] Api di primo livello di react - react.purecomponent. https://it.reactjs.org/docs/react-api.html#reactpurecomponent.

Ringraziamenti

Ringrazio la relatrice Catia Prandi e la correlatrice Chiara Ceccarini per la disponibilità dimostrata e i suggerimenti proposti durante lo svolgimento del lavoro di tesi.

Ringrazio infine tutti coloro che mi hanno sostenuto durante questo percorso universitario e che hanno creduto nel raggiungimento di questo mio importante traguardo.