

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

LOREM IPSUM
DOLOR SIT AMET
EXPECTO TITOLORUM
(TODO: alla fine)

Relatore:
Prof. Giovanni Delnevo

Presentata da:
Francesco Valentini

Correlatore:
Dott. Kelvin Olaiya

Sessione III
Anno Accademico 2023/2024

Introduzione

TODO

Indice

Introduzione	i
1 Contesto	1
1.1 Open Data	1
1.1.1 Concetto di Open Data	2
1.1.2 Caratteristiche principali degli Open Data	3
1.1.3 Formati principali	4
1.1.4 Contesto italiano e internazionale	8
1.2 Bologna e progetto BolognaWiFi	11
1.2.1 Bologna Smart City	11
1.2.2 Progetto BolognaWiFi	14
1.3 Visualizzazione dei dati	18
1.3.1 Importanza della visualizzazione	19
1.3.2 Caratteristiche di una buona visualizzazione	21
1.4 Sfide e opportunità	26
1.4.1 Formati Open File	26
1.4.2 Gestione di dataset eterogenei	27
1.4.3 Aspetti etici e legali	29
1.4.4 Opportunità	31
2 Tecnologie Utilizzate	33
2.1 HTML	33
2.2 CSS	35

2.3	SCSS	37
2.4	JavaScript	38
2.4.1	AJAX	40
2.4.2	Framework e librerie	41
2.5	Leaflet	42
2.6	Vue.js	43
2.7	NPM	44
2.8	PHP	45
2.9	API	47
2.9.1	Tipologie di API	47
2.9.2	RESTful APIs	49
2.9.3	Utilizzo delle API	49
2.9.4	Documentazione	51
2.9.5	Evoluzione delle API	52
2.10	JSON	52
2.11	SQL	53
2.11.1	MySQL	53
2.11.2	Struttura dei dati relazionali	54
2.11.3	Utilizzo nelle applicazioni web	56
2.11.4	Normalizzazione dei dati	56
2.12	Apache HTTP Server	57
2.13	XAMPP	60
2.13.1	Componenti	60
2.14	Git	62
2.14.1	Vantaggi di Git	64
2.14.2	Principali comandi di Git	66
2.14.3	Piattaforme di hosting per repositories	66
3	Visualizzazione dati sull'utilizzo del progetto BolognaWiFi	67
3.1	Il caso studio: dati del BolognaWiFi	67
3.1.1	Spostamenti	68
3.1.2	Affollamento e Affluenza	69

3.2	Struttura del database	72
3.2.1	Aree	72
3.2.2	Spostamenti	72
3.2.3	Affollamento e Affluenza	73
3.2.4	Schema finale	74
3.3	Connessione tra frontend e backend	77
3.4	Single-Page App	78
3.5	Implementazione richieste AJAX	78
3.6	Popolamento del database	79
3.7	Informazioni su una zona o spostamento	81
3.8	Calcolo del colore di una zona	81
3.9	Raggruppamento dei dati giornalieri	81
3.10	Questionario SUS sull'esperienza utente	81
	Bibliografia	81

Elenco delle figure

1.1	Open Data su Google Trends	2
1.2	Dataviz: vendite di console Nintendo Switch	22
3.1	Struttura dei dati sugli spostamenti	68
3.2	Struttura dei dati sull'affollamento	70
3.3	Struttura dei dati sull'affluenza	71
3.4	Schema del database	76
3.5	Configurazione delle impostazioni Vite	77
3.6	Funzione AJAX per ottenere la lista dei movimenti di una specifica data	79
3.7	80

Elenco delle tabelle

Contesto

In questo capitolo andremo a esplorare alcune delle tecnologie emergenti nel settore digitale che stanno inesorabilmente trasformando il mondo che ci circonda, lasciando un deciso impatto sulla realtà. Vedremo i soggetti chiave di questo cambiamento, sia a livello globale parlando di *Open Data* in generale, sia trattando gli impatti che questi ultimi stanno lasciando a livello locale sul nostro territorio, con progetti come *BolognaWiFi* del Comune di Bologna. Quindi, solleviamo la questione di come questi dati possano essere visualizzati per riuscire a trarne il maggior beneficio, e infine analizzeremo gli impatti di queste iniziative sulla realtà, considerandone sia le sfide che le opportunità offerte. Approfondiremo in sequenza ciascuna di queste sezioni, per riuscire ad avere una comprensione completa di come queste tecnologie stanno lasciando un impatto sulle nostre città.

1.1 Open Data

Le nuove tecnologie permettono di creare servizi in grado di migliorare la vita dei cittadini e di far funzionare più efficientemente governi e società. Molti dei dati necessari a raggiungere questi obiettivi sono prodotti da organismi pubblici, tuttavia spesso tali dati non sono disponibili in formati che li rendano facili da manipolare. Per questo motivo, da diversi anni si utilizza la nozione di *open data*, i cosiddetti dati aperti, e più specificamente gli *open government data*, ovvero quelli riguardanti il settore pubblico del governo,

intesi come informazione, pubblica o privata, accessibile e riutilizzabile da chiunque e per qualsiasi fine.

Come possiamo vedere in Figura 1.1, questo termine entra nell'uso comune a partire dal 2009, quando diversi governi hanno annunciato nuove iniziative per l'apertura della loro informazione pubblica, tra cui quelli di Stati Uniti, Regno Unito, Canada e Nuova Zelanda [1].

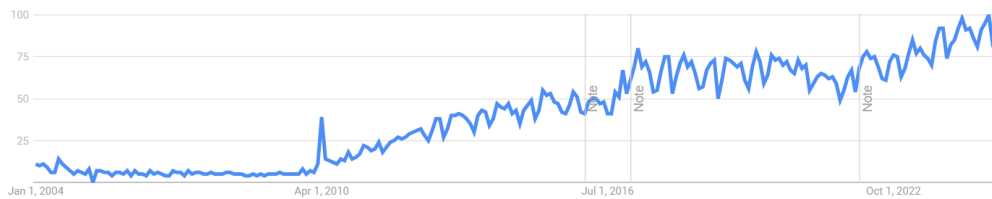


Figura 1.1: Ricerche su Google relative al termine Open Data effettuate dal 2004 al 2025. Notare l'aumento successivo al 2009 e il trend in crescita [2].

1.1.1 Concetto di Open Data

Gli *open data*, specialmente quelli della pubblica amministrazione, costituiscono una formidabile risorsa che ancora oggi rimane in larga parte inutilizzata. Molte persone e organizzazioni raccolgono una vasta gamma di svariati tipi di dati per svolgere le proprie attività. In questo ambito, il governo riveste particolare importanza, sia per la quantità e la centralità dei dati che raccoglie, sia perché la maggior parte dei dati del governo sono pubblici per legge, quindi potrebbero diventare aperti e resi disponibili per l'utilizzo da parte di chiunque. Questo riveste particolare interesse in quanto si ritiene che gli *open data* possano apportare benefici in molti settori, ed esistono altrettanti esempi di come siano già stati usati con successo. Ci sono inoltre svariati gruppi di persone e organizzazioni che possono trarre vantaggi dalla disponibilità di *open data*, incluso il governo stesso. Allo stesso tempo, è impossibile effettuare previsioni esatte su come e dove verranno appor-

ti questi benefici in futuro, perché per natura dell'innovazione gli sviluppi partono spesso dai posti più inaspettati [3].

1.1.2 Caratteristiche principali degli Open Data

In base a quanto stabilito dalla *Open Definition*, gli *open data* sono dati che possono essere utilizzati liberamente e ridistribuiti da chiunque e sono sottoposti, al massimo, al riconoscimento obbligatorio di una potenziale attribuzione o crediti in generale [4].

Nella *full Open Definition* vengono specificate in dettaglio le proprietà più importanti, ovvero:

- **Disponibilità e Accesso:** i dati devono essere disponibili per intero, senza pagare più di un costo ragionevole di riproduzione, preferibilmente scaricandoli dalla rete Internet. I dati devono, inoltre, essere disponibili in un formato conveniente e modificabile.
- **Riutilizzo e Ridistribuzione:** i dati devono essere forniti sotto licenza che ne permetta liberamente il riutilizzo e la ridistribuzione, incluso specialmente il mescolamento con altri *dataset*.
- **Partecipazione Universale:** tutti devono essere in grado di utilizzare, riutilizzare e ridistribuire i dati, senza discriminazioni contro campi di applicazione o contro persone o gruppi particolari. Ad esempio, non sono permesse le licenze ristrette ai soli scopi non-commerciali, che impedirebbero l'utilizzo commerciale, e nemmeno qualsiasi altra restrizione all'uso per certe finalità, come quella esclusivamente educativa.

Queste regole sono necessarie per garantire chiarezza sul significato di apertura e, soprattutto, per assicurarne l'*interoperabilità*. Quest'ultima denota la capacità di lavorare insieme, ovvero interoperare, tra diversi sistemi e organizzazioni, che in questo caso si riferisce all'interoperabilità, detta anche mescolanza, tra dataset differenti. È una capacità molto importante perché

permette a diversi componenti di collegarsi per lavorare insieme e questo è essenziale per realizzare sistemi grandi e complessi, la cui costruzione altrimenti sarebbe quasi impossibile senza interoperabilità [5].

Il principio cardine di un *commons* di dati o codice, ovvero una raccolta di essi, è che un qualunque pezzo di materiale aperto il contenuto possa essere mescolato liberamente con altro materiale aperto. Questa interoperabilità è assolutamente necessaria a livello pratico per realizzare i principali benefici ottenibili dall'apertura dei dati, ovvero l'abilità notevolmente migliorata di combinare insieme diversi dataset e di conseguenza la capacità di sviluppare un numero maggiore di prodotti e servizi aventi qualità migliore. Questa definizione data riguardo all'apertura assicura che, dati due dataset aperti ottenuti da due fonti diverse, possiamo essere sempre in grado di combinarli insieme, evitando di trovarci di fronte a un numero enorme di dataset aventi scarsa o nulla capacità di essere mescolati insieme in sistemi di più larga scala, senza quindi avere la capacità di apportare valore aggiunto. Il punto chiave di aprire i dati deve essere il *focus* su dati non-personali, ovvero i dati non devono contenere informazioni riguardanti persone specifiche. Inoltre, per alcuni tipi di dati ottenuti dal governo, potrebbero applicarsi restrizioni nazionali per motivi di sicurezza [3].

1.1.3 Formati principali

Gli open data possono essere utilizzati in svariati formati, di seguito vediamo quali.

JSON

Quello più utilizzato in ambito software è JSON, un semplice formato file che è molto facile da leggere per qualsiasi linguaggio di programmazione. La sua semplicità lo rende più veloce da processare per un computer rispetto ad altri formati come XML.

XML

XML è un formato molto utilizzato per lo scambio di dati, in quanto fornisce buone opportunità di mantenere la struttura dei dati e il modo in cui i file sono costruiti, permettendo agli sviluppatori di scrivere parti della documentazione insieme ai dati senza interferire con la loro lettura.

RDF

RDF è un formato raccomandato dal W3C e rende possibile rappresentare i dati in una forma in cui sono più facili da combinare con altri dati provenienti da più fonti. Dati RDF possono essere salvati in varie serializzazioni, tra cui XML e JSON. RDF incoraggia l'utilizzo di URL come identificatori, fornendo un modo conveniente di interconnettere direttamente iniziative open data già esistenti sul Web. Non è ancora molto diffuso, ma è diventato un trend tra le iniziative di Open Government, tra cui annoveriamo i progetti di Open Data collegati tra i governi di Regno Unito e Spagna. L'inventore di Internet, Tim Berners-Lee, ha recentemente proposto uno schema chiamato *fivesstar* che include i dati RDF collegati, ponendolo come obiettivo da raggiungere per le iniziative di open data.

Spreadsheet

Sono utilizzati da molte autorità, le quali hanno ancora delle informazioni rimaste in documenti come quelli di Microsoft Excel. Questi dati possono essere utilizzati immediatamente con la corretta descrizione del significato delle varie colonne. Tuttavia, in alcuni casi ci possono essere delle macro e delle formule nei fogli di calcolo, che possono diventare potenzialmente più complicate da gestire. Di conseguenza, è consigliabile documentare questo genere di calcoli di fianco allo *spreadsheet*, dato che è generalmente più accessibile alla lettura da parte degli utenti.

Comma-Separated Values

I file CSV possono essere un formato molto utile data la loro compattezza e possono quindi essere adatti al trasferimento di grandi quantità di dati all'interno della stessa struttura. Tuttavia, questo formato è talmente spartano che i dati sono spesso inutili senza appropriata documentazione, dato che può essere quasi impossibile indovinare il significato delle diverse colonne. Diventa quindi particolarmente importante che la documentazione dei singoli campi sia accurata. Inoltre, è essenziale rispettare la struttura del file, in quanto la singola omissione di un campo è in grado di disturbare la lettura di tutti i rimanenti dati del file senza avere una reale opportunità di aggiustamento, perché non si può determinare come vanno interpretati i dati rimanenti.

Documento di testo

I documenti classici in formati come Word, ODF, OOXML o PDF potrebbero bastare per mostrare alcuni tipi di dati, come delle *mailing lists* relativamente stabili o equivalenti, senza richiedere un grande sforzo. Questo formato non offre alcun supporto per il mantenimento di una struttura coerente, il che spesso significa che diventa difficile inserire dati mediante strumenti automatizzati. Bisogna assicurarsi di utilizzare dei *templates* come base dei documenti che mostreranno dati per il riuso, così da rendere possibile ottenere informazioni da tali documenti. Inoltre, per supportare ulteriormente l'utilizzo dei dati si potrebbero utilizzare dei *markers* tipografici il più possibile, così da rendere più facile per una macchina la distinzione degli *headings*, ovvero i titoli di qualsiasi tipo, dal contenuto, e così via. Generalmente è raccomandabile non esibirli in un formato di *word processing*, se i dati esistono in formati differenti.

Testo non formattato

I documenti di testo non formattato (.txt) sono molto semplici da leggere per un computer, ma generalmente escludono metadati strutturali dall'interno del documento, quindi gli sviluppatori devono creare un *parser* che possa interpretare ogni documento nell'ordine in cui appare. Possono verificarsi alcuni problemi dallo scambio di file *plain text* tra diversi sistemi operativi, in quanto Windows, macOS e altre varianti Unix hanno ciascuno il loro modo di indicare al computer che si è raggiunta la fine della riga, quindi 3 diversi caratteri di *newline*.

Immagini scannerizzate

Probabilmente questo è il metodo meno indicabile per la maggioranza dei dati, ma sia TIFF che JPEG-2000 sono in grado di dargli una documentazione di ciò che si trova nell'immagine, e possono addirittura indicare testualmente l'intero contenuto del documento all'interno di un'immagine dello stesso documento. Potrebbe essere rilevante mostrare dati come immagini i cui dati non sono stati creati elettronicamente, un esempio sono i materiali di archivio, per i quali avere un'immagine è sicuramente meglio di nulla.

Formati proprietari

Alcuni sistemi dedicati hanno i propri formati di dati con cui possono essere salvati o esportati. Talvolta può bastare esporre i dati in tale formato, specialmente se ci si aspetta che un ulteriore utilizzo avvenga in un sistema simile a quello da cui provengono. Va sempre indicato dove si possono trovare ulteriori informazioni su questi formati proprietari, per esempio fornendo un link al sito web del fornitore. Generalmente, se fattibile, è raccomandabile mostrare i dati in formati non-proprietari.

HTML

Al giorno d'oggi, molti dati sono disponibili in formato HTML su vari siti e questo potrebbe tranquillamente essere sufficiente se i dati sono molto stabili e con ambito limitato. In alcuni casi, però, potrebbe essere preferibile avere i dati in una forma più facile da scaricare e manipolare, ma dato che il riferimento a una pagina su un sito web è una modalità molto semplice ed economica, potrebbe costituire un buon punto di partenza nella visualizzazione di dati. Tipicamente, sarebbe più appropriato utilizzare delle tabelle in documenti HTML per contenere i dati, ed è importante che i vari campi di dati siano visualizzati e assegnati degli ID che li rendano facili da trovare e manipolare i loro dati. Per effettuare lo scraping, Yahoo ha sviluppato un *tool* in grado di estrarre informazioni strutturate da un sito, e tali strumenti sono in grado di effettuare molte altre operazioni purché i dati siano accuratamente etichettati [6].

1.1.4 Contesto italiano e internazionale

Ad oggi, è già possibile indicare un cospicuo numero di settori nei quali i dati aperti del governo stanno già apportando un certo valore, tra cui:

- Controllo di trasparenza e democrazia.
- Partecipazione.
- Autopotenziamento.
- Miglioramento o creazione di nuovi prodotti e servizi da parte di privati.
- Innovazione.
- Miglioramento dell'efficienza di servizi del governo.
- Miglioramento dell'efficacia di servizi governativi.
- Misurazione dell'impatto di certe politiche.

- Nuova conoscenza ottenuta dalla combinazione di fonti di dati e dalla ricerca di *pattern* all'interno di grandi volumi di dati.

Riguardo alla trasparenza, esistono progetti come il finlandese *tax tree*, l'albero delle tasse, e il britannico *where does my money go*, dove vanno i miei soldi, per visualizzare in che modo i ricavi derivanti dalle tasse vengono spesi dal governo. Inoltre, possiamo ricordare come gli open data abbiano risparmiato al Canada \$3.2 miliardi in una frode fiscale relativa a un ente di beneficenza. Altri siti web includono il danese *folketsting.dk* per tracciare l'attività in parlamento e i processi di creazione di leggi, così da poter vedere esattamente gli eventi che stanno accadendo, e quali parlamentari sono coinvolti.

Gli open data del governo possono anche aiutarci a prendere decisioni migliori nella nostra vita, o a diventare più attivi nella società. Una donna in Danimarca ha sviluppato *findtoilet.dk*, che mostra tutti i bagni pubblici danesi così che le persone che conosceva con problemi alla vescica potessero diventare più sicuri di se stessi quando andavano nuovamente fuori. In Olanda è disponibile un servizio, *vervuilingsalarm.nl*, che ci avvisa con un messaggio se domani la qualità dell'aria nelle vicinanze raggiungerà una soglia che abbiamo definito in precedenza. A New York possiamo facilmente capire dove possiamo portare il cane a passeggio, insieme a trovare altre persone che frequentano gli stessi parchi. Servizi come *Mapumental* nel Regno Unito e *Mapnificent* in Germania permettono di trovare dei posti in cui vivere, prendendo in considerazione la durata del tragitto per arrivare a lavoro, il prezzo delle case e la qualità di una certa zona. Tutti questi servizi, per funzionare, si appoggiano a dati forniti apertamente dal governo.

Economicamente, gli open data sono altrettanto importanti: diversi studi hanno stimato il valore economico degli open data in alcune decine di miliardi di Euro su base annua, e questo solamente nell'Unione Europea. Nuovi prodotti e aziende stanno riutilizzando dati aperti, come ad esempio il danese *huusetsweb.dk* che aiuta a trovare dei modi per migliorare l'efficienza energetica della propria casa, insieme alla pianificazione finanziaria e alla ricerca

di costruttori che possano svolgere il lavoro. Tutto è basato sul riutilizzo delle informazioni catastali e informazioni sui sussidi da parte del governo, insieme al registro commerciale locale. Google Traduttore utilizza un enorme quantità di documenti dell'UE che compaiono in tutte le lingue europee per addestrare gli algoritmi di traduzione e quindi migliorare la sua qualità del servizio.

Gli open data offrono del valore anche al governo stesso, ad esempio per aumentarne l'efficienza. Il Ministero Olandese dell'Istruzione ha pubblicato online tutti i suoi dati relativi all'istruzione, permettendone il riuso. Da allora, il numero di domande che hanno ricevuto è crollato, riducendo il carico di lavoro e i relativi costi, mentre per i dipendenti pubblici è più facile rispondere alle rimanenti domande, in quanto diventa maggiormente chiaro dove possono essere trovate le informazioni rilevanti. Gli open data hanno anche reso più efficiente la pubblica amministrazione, il che porta a un'ulteriore riduzione dei costi. Il dipartimento olandese dei beni culturali sta rilasciando attivamente i loro dati e collabora con società amatoriali di storia e gruppi come la *Wikimedia Foundation* per eseguire i loro compiti in maniera più efficiente. Questo non solo porta a miglioramenti nella qualità dei loro dati, ma contribuisce anche a rimpicciolire le dimensioni del dipartimento stesso.

Anche se ci sono già numerose istanze relative ai modi in cui gli open data stanno creando del valore aggiunto sia in ambito sociale che in quello economico, non sappiamo ancora nulla riguardo alle nuove cose che diventeranno possibili. Nuove combinazioni di dati possono creare ulteriore conoscenza, il che può potenzialmente portare a scoprire campi di applicazione completamente nuovi. Lo abbiamo già visto in passato, per esempio quando nel XIX secolo a Londra il Dott. Snow ha scoperto il legame tra inquinamento dell'acqua potabile e colera, combinando i dati delle morti attribuibili al colera con la posizione dei pozzi d'acqua. Questo portò alla costruzione dei sistemi fognari di Londra e migliorò notevolmente la salute pubblica della popolazione. Probabilmente potremmo vedere accadere nuovamente degli

sviluppi simili, scaturiti dalla combinazione di diversi dataset aperti. Questo potenziale, ancora non sfruttato, può essere rilasciato se trasformiamo i dati pubblici del governo in open data, ma solo se sono davvero aperti, ovvero se non sottoposti ad alcun tipo di restrizione legale, finanziaria o tecnologica relativa al suo riutilizzo da parte di terzi. Ogni restrizione impedisce alle persone di riutilizzare i dati pubblici, o quantomeno lo rende più difficile. Per poter realizzare appieno il loro potenziale, i dati pubblici devono essere aperti, ovvero degli open data [3].

1.2 Bologna e progetto BolognaWiFi

Le Smart Cities sono aree urbane intelligenti e sostenibili, in grado di pianificare in modo coerente l'integrazione delle diverse caratteristiche dell'identità del proprio territorio, siano esse economiche, produttive o ambientali, in un'ottica di innovazione.

Bologna ha scelto di intraprendere questo percorso per sviluppare soluzioni utili ad affrontare problematiche urbane e sociali, mettendo la tecnologia al servizio delle persone, attraverso un'alleanza tra mondo della ricerca, Università, imprese e pubblica amministrazione.

Basandosi su questa definizione, la città di Bologna vuole essere pensata come un sistema intelligente e sostenibile dove si sfrutta l'ottimizzazione delle risorse per qualificare i servizi già esistenti e crearne di nuovi per costruire una città aperta alla partecipazione e al contributo creativo dei cittadini, ovvero il *civic commons*, così come previsto dal Piano Strategico Metropolitano.

Questo accordo sottoscritto ha anche lo scopo di implementare le attività svolte finora da Comune, Alma Mater e Aster sul tema delle Smart Cities ed è aperto all'adesione di altri enti e imprese [7].

1.2.1 Bologna Smart City

Il progetto di Bologna Smart City è iniziato il 30 luglio 2010, quando il Comune di Bologna, l'Alma Mater e Aster hanno firmato un Protocollo

d'intesa per la realizzazione della relativa piattaforma, avente l'obiettivo di rispondere ai bisogni della comunità per migliorarne la qualità della vita e garantire i diritti fondamentali della socialità, dell'istruzione, dello sviluppo e della salute.

Una città è intelligente se compie delle scelte nette e sostenibili, per cui bisogna investire con azioni strategiche nel campo di energia, servizi, digitale e valorizzazione dei beni ambientali e culturali. In questo ruolo, l'Università mette a disposizione i propri saperi, consapevole del suo legame storico con Bologna, figurandosi come un grande consulente sullo sviluppo della città, della società e dell'impresa. Bologna Smart City è un obiettivo prioritario affinché la regione Emilia-Romagna sia in grado di restare al passo con i tempi, in cui l'innovazione è fondamentale per la competitività imprenditoriale e per il sistema della ricerca, proprio a partire dalla rete di alta tecnologia.

In questo progetto, i tre partner (Comune, Università e Aster) hanno individuato un primo gruppo di 7 ambiti chiave su cui verteranno le azioni sui temi principali:

- Beni Culturali: valorizza e riqualifica il centro storico e il suo patrimonio culturale, insieme ai portici e al turismo.
- Iperbole 2020 Cloud & Crowd: riprogetta la Rete Civica Iperbole, basandosi sulla tecnologia cloud e un'identità digitale integrata, raccogliendo così l'offerta di contenuti e servizi di Pubblica Amministrazione, imprese e cittadini.
- Reti intelligenti: smart grid, banda ultralarga FTTH e smart lighting.
- Mobilità sostenibile: sviluppo di una rete della mobilità elettrica intelligente.
- Quartieri sicuri e sostenibili: ristrutturazione il patrimonio pubblico e privato per aumentare l'efficienza e la produzione di energia, monitora la sicurezza degli edifici, gestisce i rifiuti, configura il social housing, la do-

motica, il co-working, insieme a servizi e nuovi ambienti per lavoratori della conoscenza e ricercatori.

- Sanità e Welfare: e-care, e-health, ottimizzazione dei processi e business intelligence.
- Educazione e istruzione tecnica: sviluppo progetti in ambito educativo, promuove una nuova cultura tecnica e scientifica [7].

Bologna è sempre di più una smart city e ha l'obiettivo di diventare una città sostenibile. Si trova già da anni ai primi posti delle classifiche delle smart city italiane. Una città che punta non solo ad essere inclusiva, sicura, attenta alla salute e all'ambiente, ma anche a fornire a tutti i suoi abitanti una serie di servizi tra cui in primis ci sono quelli della mobilità. Sappiamo che l'inquinamento atmosferico è un serio problema per la salute delle persone ed è legato alle emissioni, quindi una città smart sostenibile dovrebbe mettere a disposizione dei cittadini tutta una serie di sistemi volti a ridurre i livelli di CO₂, particolato e altre sostanze pericolose.

Nasce quindi il concetto di MaaS, Mobility as a Service, che consiste in una nuova idea di mobilità in cui si prevede l'integrazione dei vari servizi di trasporto pubblico e privato in un unico canale digitale, rendendoli più accessibili all'utente finale, e questa sharing mobility è già in sperimentazione a Milano. In questo modo, i cittadini possono scegliere tra varie soluzioni di mobilità sostenibile, configurandole come un'alternativa all'utilizzo dell'auto privata.

A Bologna ne sono già un esempio il servizio di car-sharing elettrico e di bike-sharing, sia con biciclette classiche che con e-bike (biciclette a pedalata assistita): tutti questi servizi sono noleggiabili tramite una semplice app. Inoltre, a Bologna è attivo il servizio di Telepass Pay, che permette tramite app di pagare il parcheggio, prenotare un taxi, o prenotare la pulizia dell'auto e della moto nel punto in cui è parcheggiata. La tecnologia svolgerà sempre di più un ruolo fondamentale nella diffusione di questi servizi, sempre più diversificati e disponibili, permettendo al cittadino di scegliere il mezzo di

trasporto più idoneo al tragitto da compiere e al contempo diminuendo il traffico in città, migliorando la qualità dell'aria e riducendo le emissioni [8].

1.2.2 Progetto BolognaWiFi

Verso la fine degli anni '90, la rete Internet era sempre più diffusa e stava iniziando a diventare un'infrastruttura comunicativa, costituendo così una grande opportunità al servizio dei cittadini. Di conseguenza alcuni comuni italiani, tra cui quello di Bologna, riuscirono a comprenderne in maniera lungimirante le grandi potenzialità che offriva per costruire nuovi spazi di informazione, relazione e connessione civica. Così nel 1995 il Comune di Bologna, in collaborazione con l'Alma Mater, creò la rete civica Iperbole, la quale venne pensata e sviluppata come una rete globale-locale. Questo progetto poté nascere proprio a Bologna in virtù di una rara situazione di privilegio riguardante l'innovazione amministrativa, molto più avanzata all'epoca rispetto alle altre città italiane. La rete Iperbole offriva agli utenti la possibilità di avere accesso a Internet, fornendo anche un indirizzo email gratuito, e in questo modo ha saputo contribuire attivamente all'alfabetizzazione digitale dei cittadini bolognesi.

Iperbole è suddivisa in tre sezioni. Nella prima, quella del Comune, vengono riportate le iniziative e le attività svolte dal Comune di Bologna, così da permettere di rimanere aggiornati sulle tematiche correlate a Sindaco, Giunta, Consiglio Comunale, Elezioni, Turismo, Mobilità, Qualità dell'aria, Casa e Comunicati. Nella seconda, quella dei Servizi Online, è possibile usufruire di tutti i servizi messi a disposizione dal Comune, come il pagamento di multe e bollettini MAV, la prenotazione per l'URP, la richiesta di un assegno familiare, la presentazione della dichiarazione ISEE e la segnalazione di problemi. Nella sezione Partecipa, invece, il cittadino può impegnarsi e partecipare attivamente ai processi di collaborazione e cura dei beni comuni. Negli ultimi anni il Comune di Bologna, in collaborazione con Lepida SpA, ha inaugurato la rete Iperbole Wireless, che offre a cittadini e visitatori un servizio di connessione a Internet tramite WiFi, che si configura come un'e-

voluzione *mobile* della Rete Civica Iperbole. Tale rete wireless è aperta e non richiede autenticazione.

Dal 1995 ad oggi, Iperbole non è più solo una rete telematica di informazione costante e aggiornata per i cittadini, ma sta diventando un vero e proprio punto di riferimento grazie alla facilità di accesso e alle iniziative di partecipazione civica, rappresentando non solo una grande invenzione tecnologica, ma soprattutto un esempio di innovazione sociale grazie al quale si sono garantiti e messi in atto diritti digitali e di cittadinanza [9].

Il Comune di Bologna, in collaborazione con Lepida spa, offre a cittadini e visitatori un servizio di connessione a internet in modalità Wi-Fi come evoluzione *mobile* della Rete Civica Iperbole. Per navigare è sufficiente collegarsi alla rete BolognaWiFi, la quale è aperta e non richiede autenticazione. Nelle aree all'aperto è possibile accedere e navigare tutti i giorni 24 ore su 24, mentre nei luoghi all'interno di edifici si può navigare negli orari di apertura delle strutture che offrono il servizio. Ad oggi ci sono più di 300 hotspot attivi in città [10].

Già nel 2014, dopo un anno intero di operatività della connessione Wi-Fi pubblica a banda ultralarga, piazza Maggiore a Bologna era riuscita a ottenere il titolo di *piazza più connessa d'Europa*, segno dell'importanza di questo progetto per la città felsinea [11].

BolognaWiFi è un servizio offerto dal Comune di Bologna e fornisce una connettività wireless gratuita a tutti i cittadini. Nato da una collaborazione con alcuni operatori nel settore delle telecomunicazioni, i primi test sono iniziati nel 2005 in Piazza Maggiore, insieme ad alcune ditte private che hanno aderito al progetto [12].

Obiettivi principali

Tra gli obiettivi, Iperbole voleva garantire lo scambio di informazioni non soltanto tra le associazioni, ma anche tra individui, istituzioni e, in generale, con chiunque nel mondo avesse a disposizione un computer e un modem, favorendo forme di partecipazione e collaborazione civica. Iperbole, tutt'oggi,

rappresenta il tentativo di riprodurre in uno spazio digitale tutto quello che è lo spazio della città, creando situazioni di incontro e di confronto nella realtà più ampia che è quella di Internet. Queste realtà virtuali, dette anche agorà elettroniche con un riferimento alla piazza principale dell'antica Grecia, intendono rappresentare un punto di riferimento per il cittadino, una specie di mappa con cui orientarsi, informarsi e partecipare alle attività pubbliche della società [9].

Lo scopo principale del progetto BolognaWiFi è quello di abbattere il *digital divide*, agevolare la connettività mediante un servizio gratuito e, allo stesso tempo, favorire i cittadini nelle più svariate attività, a prescindere dal fatto che siano residenti, studenti, pendolari o turisti. L'idea originale era che si potesse navigare gratuitamente, sfruttando una miriade di utilità, tutto rimanendo seduti comodamente a uno dei bar del centro o all'ombra delle Due Torri [12].

Tipologie di dati raccolti

Il servizio BolognaWiFi, come gli altri servizi di EmiliaRomagnaWiFi, nel corso del suo funzionamento raccoglie automaticamente alcuni dati personali dell'utente necessariamente correlati alle attività tecniche strumentali alla resa del servizio in questione. In particolare, sono trattati:

- Timestamp di richiesta o di rinnovo o di cessazione dell'indirizzo IP alla rete da parte del dispositivo mobile.
- Indirizzo IP intranet assegnato al dispositivo mobile dalla rete.
- Identificativo MAC address del dispositivo mobile immediatamente pseudonimizzato con opportuni algoritmi non invertibili.

L'indirizzo IP intranet assegnato viene utilizzato in tutti i punti di accesso che fanno capo ad una stessa rete. La dimensione di una rete varia a seconda dei territori, ma tipicamente è costruita in modo da includere almeno tre diversi punti di accesso. Il trattamento dei dati è necessario per l'esecuzione

di compiti di interesse pubblico o connessi all'esercizio di pubblici poteri di cui è investita la regione Emilia-Romagna e non necessita di un consenso. Nello specifico, all'interno del servizio BolognaWiFi (EmiliaRomagnaWiFi) i dati personali acquisiti vengono trattati per lo svolgimento di funzioni istituzionali e, in particolare, è finalizzato a offrire il servizio di WiFi a banda ultra larga, pubblico, libero, senza autenticazione e gratuito, monitorandone il corretto funzionamento al fine di assicurarne qualità, continuità e accessibilità. I dati personali acquisiti sono trattati, con strumenti informatici, rispettando i principi stabiliti dalla normativa vigente, attraverso modalità idonee a garantire la riservatezza e la sicurezza dei dati. In particolare, vengono rispettati i principi di liceità, correttezza, trasparenza, limitazione delle finalità, minimizzazione dei dati, esattezza, limitazione della conservazione, integrità e riservatezza, e responsabilizzazione stabiliti dal GDPR. Il trattamento di aggregazione ed elaborazione per fini statistici viene effettuato su dati pseudonimizzati e qualsiasi statistica puntuale viene prodotta se coinvolge almeno tre diversi dispositivi mobili. I dati raccolti non consentono di identificare la posizione del dispositivo mobile tra i punti di accesso che fanno parte della stessa rete. Diventa quindi impossibile procedere a una reidentificazione dell'interessato, a meno che tale soggetto non lo richieda direttamente, indicando il proprio MAC address. Anche in questo caso, ciò è possibile solamente se tale indirizzo non è stato rigenerato casualmente dal sistema operativo del dispositivo mobile. I dati personali non sono oggetto di comunicazione o diffusione e non sono mai trasferiti al di fuori dell'Unione Europea. Rimane il fatto che il conferimento dei dati non sia obbligatorio [13].

A partire da questi dati grezzi vengono poi ricavati dati più elaborati in forma aggregata su base giornaliera, come:

- Matrice spostamenti: totale degli spostamenti orari delle persone da e verso ciascuna zona coperta da WiFi pubblico [14].
- Affluenza: viene registrato il numero di dispositivi che sono entrati in una data area coperta da WiFi pubblico, per ogni ora del giorno [15].

- Affollamento: viene registrato il numero di dispositivi che sono presenti in una certa zona coperta da WiFi pubblico, per ogni ora del giorno [16].
- Connessioni giornaliere: viene registrato il totale di connessioni giornaliere al servizio di WiFi pubblico BolognaWiFi. Uno stesso dispositivo che si collega più volte al servizio nell'arco di 24 ore viene contato una volta sola [17].

Oltre a questi dati, vengono salvati anche l'elenco di hotspot [10] assieme all'elenco di aree coperte da segnale del BolognaWiFi [18].

1.3 Visualizzazione dei dati

La *data visualization* consente alle Città di comunicare facilmente con i Cittadini, visualizzando le città e le loro dinamiche attraverso strumenti capaci di trasformarle in rappresentazioni grafiche o schematiche. In questo modo, si offrono ai cittadini informazioni rapide ed esaustive per costruire un dialogo con l'amministrazione, facendo della Data Visualization una grande opportunità per città sempre più smart. Nel mondo, il 90% dei dati circolanti è stato generato negli ultimi due anni. Ogni giorno generiamo globalmente 2.5 EB (Exabytes) di nuovi dati e colleghiamo alla rete circa 1000 miliardi di dispositivi e oggetti a livello globale, creando un ambiente davvero favorevole all'Internet of Things (IoT). La data visualization, detta anche visualizzazione dati, è la rappresentazione dei dati in formato grafico o comunque visivo, la quale permette di analizzare grandi quantità di dati, ovvero i Big Data, in modo semplice, tramite grafici e rappresentazioni più comprensibili di una lista grezza di dati. Data la nostra scarsissima soglia di attenzione, che secondo gli ultimi rilievi si attesta a circa 8 secondi, questa tecnica diventa importantissima data la grande disponibilità di informazioni che possediamo. Le rappresentazioni grafiche più semplici sono anche quelle più riuscite, e al contempo le più potenti [19].

1.3.1 Importanza della visualizzazione

Le nostre città generano continuamente dei dati, basta considerare la tradizionale urbanistica, i flussi di traffico, la disponibilità di parcheggio, la qualità dell'aria, la gestione energetica e la sensoristica avanzata. I sistemi informativi territoriali offrono da molto tempo un sistema per analizzare velocemente questi dati, capaci di evolvere velocemente nel tempo e fornire agli amministratori delle informazioni in modo rapido ed efficace. Tra tutte queste tecniche di elaborazione di dati e visualizzazione, alcune sono ormai canonizzate e considerabili *commodity*, ma dall'arrivo di Google Maps e della presenza ubiquitaria di strumenti di localizzazione abbiamo una nuova ondata di *location awareness* da parte degli utenti finali, siano essi cittadini residenti, turisti o businessmen, che può essere utilizzata dalle amministrazioni per migliorare il dialogo tra pubblico e privato.

Per esempio, affermare che l'impatto del cambiamento climatico stia rendendo più complessa la vivibilità delle città, o che entro 30 anni le città accoglieranno più di 2.5 miliardi di persone, è diverso dal mostrare una mappa della stessa città con l'andamento delle temperature medie nel tempo per ogni zona, o una mappa dell'Italia avente i flussi migratori in entrata e uscita, sia interni che esterni.

Riguardo alle applicazioni *business to business*, o B2B, la data visualization ha un forte impatto in quanto permette di analizzare i fenomeni in modo multidimensionale dove la localizzazione geografica può avere un forte impatto e consente quindi la comprensione di situazioni che sarebbero difficili da analizzare mediante una visione tabellare del fenomeno.

Inoltre, per via del modo in cui il cervello umano elabora le informazioni, per visualizzare grandi moli di dati è più efficace e comprensibile impiegare mappe, grafici, illustrazioni o combinazioni di questi, rispetto all'utilizzo di fogli di calcolo o report. La data visualization, oltre alla facilità di comprensione, permette:

- Riconoscimento di pattern nei dati in modo intuitivo e veloce.

- Rapida identificazione degli outlier, ossia gli elementi che non seguono nessun trend particolare.
- Comprensione di fattori che influenzano i fenomeni.
- Predizione di fenomeni e trend [19].

Si dice che *un'immagine vale più di mille parole* e oggi, nell'era dei big data, con le aziende che vengono inondate da informazioni provenienti da vari tipi di dati e fonti on-premise e cloud-based, quel detto non potrebbe essere più vero. Filtrare le informazioni per distinguere quelle utili da quelle inutili sta diventando un compito sempre più arduo, ma le visualizzazioni rendono l'analisi molto più semplice e veloce, in quanto basta una sola occhiata per trovare le informazioni utili. Inoltre, noi umani sappiamo rispondere molto meglio alle visualizzazioni rispetto che al testo, dato che il 90% delle informazioni inviate al cervello è di tipo visivo e il cervello elabora gli elementi visivi 60 000 volte più velocemente rispetto a quelli testuali. Di conseguenza, capiamo come la visualizzazione dati possa offrire benefici per l'analisi e la trasmissione di informazioni.

La visualizzazione dei dati genera impatto, fa parte di molti strumenti di business intelligence, ed è fondamentale per ottenere Advanced Analytics. Permette alle persone di trovare un senso in tutte le informazioni o dati generati oggi. Grazie alla data visualization, le informazioni vengono rappresentate in forma grafica, mediante un grafico a torta, un diagramma o altri tipi di presentazione visuale.

I visual analytics sono importanti perché una buona visualizzazione dei dati è fondamentale per poterli analizzare e prendere decisioni mirate, permettendo così alle persone di vedere e comprendere in modo semplice e rapido dei modelli e relazioni, individuando trend emergenti che potrebbero passare inosservati se presentati solo in una tabella o in un foglio di calcolo contenente numeri non elaborati. Generalmente non è richiesta alcuna formazione specializzata per interpretare il contenuto dei grafici, quindi possiamo ottenere una comprensione universale. Un grafico ben progettato non offre solo

informazioni, ma riesce anche ad aumentare l'impatto di quest'ultime grazie a una presentazione forte, che attira l'attenzione e suscita interesse in un modo che nessuna tabella o foglio di calcolo sarebbe in grado di fare.

La maggior parte degli strumenti di visualizzazione dei dati è in grado di connettersi con fonti di dati, come i database relazionali. Questi dati possono essere archiviati on-premise o nel cloud e vengono recuperati per l'analisi. Gli utenti possono selezionare il modo migliore per presentare i dati da numerose opzioni. Alcuni strumenti forniscono automaticamente consigli di visualizzazione in base al tipo di dati presentati.

Un grafico dovrebbe prendere sempre in considerazione il tipo e lo scopo dei dati, in quanto alcune informazioni si adattano meglio a un tipo di grafico rispetto che a un altro, per esempio un grafico a barre invece di un grafico a torta. Con la maggior parte degli strumenti, l'utente ha una vasta scelta di opzioni di analytics visuali, da diagrammi comuni come grafici a linee e a barre, fino a sequenze temporali, mappe, tracciati, istogrammi e design personalizzati.

La visualizzazione dei dati può aiutare a raccontare la storia, trasmettendo in modo chiaro problemi complessi. Può svolgere un ruolo chiave nell'identificazione delle informazioni significative estraendole dal rumore, includendo sia anomalie che eccezioni. Inoltre, è in grado di aiutare con il crescente volume di dati, dato che l'interazione visuale con grandi dataset può semplificare l'analisi, rivelando nuovi insight. Tutto questo è permesso dalla visualizzazione dei dati, ma è necessario scegliere lo strumento giusto [20].

1.3.2 Caratteristiche di una buona visualizzazione

La disciplina della Visualizzazione Dati è in continua evoluzione, dato l'uso sempre più ampio in un numero di campi sempre maggiore. All'inizio era solamente possibile trasformare i dati in grafici statici, ma oggi l'evoluzione ha portato a possibilità straordinarie. Per costruire una buona visualizzazione dati, ci sono tre aspetti fondamentali:



Figura 1.2: Quando pensiamo alla Nintendo Switch TM, la prima cosa che viene in mente è la console stessa. Tuttavia, i dati ci mostrano che l'89% delle vendite nel 2020 hanno riguardato software compatibile con la console, e non il relativo hardware [21].

1. La visualizzazione deve essere efficace.
2. Deve evidenziare ed esplicitare i dati e le connessioni tra dati troppo difficili da spiegare con sole parole.
3. Deve rendere facilmente comprensibili a chiunque sia le informazioni presentate, sia le possibili soluzioni al problema che si sta presentando.

Ne è un esempio la seguente visualizzazione in Figura 1.2 relativa alle vendite della console Nintendo Switch TM nel corso del tempo, ripartite tra software e hardware.

Queste regole di base sono molto generiche, ma devono restare una necessità sia quando di dati da gestire e presentare sono tanti, sia quando sono

pochi. La complessità del lavoro non deve far desistere dal seguire queste linee guida, al contrario, maggiore è la mole di dati e più si rivela necessario seguire queste regole per arrivare a un buon risultato.

La prima cosa da fare quando si pensa a una visualizzazione dati è capire esattamente qual è il messaggio che vogliamo trasmettere, e con quali dati. Inoltre, è molto importante avere chiaro qual è il tipo di utente che utilizzerà la nostra dashboard, per evitare di presentare informazioni giuste alle persone sbagliate, e viceversa. Per esempio, all'interno di un'azienda è molto difficile il direttore finanziario sia interessato ai *conversion rate* delle ultime campagne di marketing, ma è molto più probabile che sia interessato alla spesa di queste campagne messa in relazione con i profitti che ha generato.

Solitamente la complessità del lavoro deriva dal numero di variabili che entrano in gioco, di conseguenza è cruciale scegliere il tipo di grafico giusto.

In generale, i grafici a torta o a ciambella sono da evitare se le variabili sono più di 3, poiché si arriva a un punto dove la visualizzazione non è più efficace e non riesce a trasmettere in modo chiaro il messaggio. Una distribuzione statistica di alcune variabili rappresentata con un tale grafico a torta, per esempio, sarebbe pressoché inutile, dato che l'utente non riuscirebbe a capire il messaggio. Al contrario, per la stessa visualizzazione della distribuzione statistica su dati numerici, sarebbe molto più adatto un istogramma, o un box plot, se abbiamo interesse nelle sue modalità e valori anomali.

Riguardo ai colori, è significativo utilizzarli nel modo appropriato: nel caso di una distribuzione statistica, in presenza di valori anomali o *outliers*, per metterli in evidenza è consigliabile utilizzare dei colori accesi, così da farli risaltare e permettergli di comunicare già dei concetti in maniera autonoma. Per esempio, un mese dell'anno avente ricavi al di sopra della media potrebbe essere evidenziato in verde, colore che richiama un'idea positiva, mentre un mese avente costi fuori dalla media potrebbe essere etichettato in rosso, per indicarne la negatività. In alternativa, potremmo anche utilizzare diverse sfumature dello stesso colore, assegnando una sfumatura intensa per i dati maggiori e una più tenue per quelli minori.

Un'ottima visualizzazione dovrebbe ridurre la complessità insita nelle tante variabili dei dati e, nel caso si dovessero compiere delle scelte, queste devono sempre aiutare il processo decisionale. In un progetto complesso invece è ancora più indicato se i dati vengono presentati in modo gerarchico, seguendo le priorità aziendali.

In generale, a prescindere dalle variabili e dalle osservazioni disponibili, basta seguire alcune regole pratiche per trasformare le visualizzazioni in un mezzo potente per illustrare il concetto alla base dei dati. Ne esistono tantissime, ma queste sono quelle principali:

- Il colore del grafico non deve essere in contrasto con il background. Lo sfondo dovrebbe essere abbastanza neutro, così da garantire la massima libertà in termini di sfumature e colori per i grafici. Bisogna inoltre tenere in mente il fatto che le dashboard potrebbero venire stampare, quindi un utilizzo eccessivo del nero sullo sfondo potrebbe essere controindicato.
- Ogni variabile dovrebbe avere dei punti di riferimento, quando necessario. Per esempio, se forniamo i dati in termini di vendita per l'anno corrente, non tutti potrebbero essere in grado di discernere se si tratti di performance buone o cattive, indipendentemente dal fatto che la curva sia in crescita o in decrescita. Se invece mostriamo anche i dati delle vendite dello scorso anno, possiamo creare un punto di riferimento in grado di aiutare chiunque a capire meglio il significato del grafico.
- I grafici a barre vanno etichettati con i numeri, ma non eccessivamente. Bisogna aggiungere solo le indicazioni strettamente necessarie per far comprendere, tenendo a mente che i numeri lunghi sono generalmente difficili da visualizzare e che, se la precisione del dato non è indispensabile, è consentito e anzi consigliato arrotondare. Ad esempio, il valore *10 523* può essere arrotondato a *10K*, incrementando la chiarezza e riducendo la complessità.

- I dati vanno ordinati per enfatizzare la scala ove possibile, ma bisogna fare attenzione a non trasmettere un messaggio sbagliato.

La regola generale è che, minore è il quantitativo di dati che utilizziamo nelle visualizzazioni, maggiore è la chiarezza di quella risultante. Bisogna lavorare con l'essenziale in termini di grafica [21].

Applicazioni pratiche: dati sul traffico in Regno Unito

Andrew Nicolson è uno sviluppatore software che è stato coinvolto in una campagna di successo contro la costruzione di una nuova strada, la *Westbury Eastern bypass*, nel Regno Unito. Nell'aprile 2010, Andrew era interessato a ottenere accesso e a utilizzare i dati sul traffico della strada che stavano venendo utilizzati per giustificare le proposte e così è riuscito a ottenere alcuni dei dati rilevanti mediante la libertà di richieste di informazioni, ma il governo locale gli ha fornito i dati in un formato proprietario che può essere letto solamente utilizzando software prodotto da un'azienda chiamata *Saturn*, che è specializzata in modellazione di traffico e sulle relative previsioni. Non esiste una versione del software di sola lettura, quindi il gruppo di Andrew non aveva altra scelta se non acquistare una licenza software, finendo per pagare £500 (€600) utilizzando uno sconto per studenti. Il pacchetto software principale veniva offerto da *Saturn* a partire da £13 000, oltre €15 000, un prezzo fuori misura per la maggioranza dei normali cittadini. Anche se nessuna legge nel diritto dell'informazione garantisce un diritto ad avere l'accesso ai dati in formati aperti, le iniziative open data del governo stanno iniziando ad essere accompagnate da documenti di *policies* che stabiliscono che le informazioni ufficiali debbano essere rese disponibili in formati di file che siano aperti. L'amministrazione Obama ha definito il *gold standard*, con la *Open Government Directive* emessa nel dicembre 2009, la quale decreta che, nella misura praticabile e soggetta a restrizioni valide, le agenzie debbano pubblicare informazioni online in un formato aperto che possa essere recuperato, scaricato, indicizzato e ricercato da applicazioni di ricerca web di utilizzo comune. Un formato aperto è inteso come uno che sia indipenden-

te dalla piattaforma, leggibile dalla macchina e reso disponibile al pubblico senza restrizioni che impediscano il riuso di tali informazioni [22].

1.4 Sfide e opportunità

Gli open data sono come una porta aperta, che invita a trovare nuove opportunità che permettano a individui, organizzazioni e comunità di esplorare, analizzare e utilizzare i dati per generare nuove idee, informazioni e soluzioni, promuovendo una fluida circolazione di informazioni, la trasparenza, l'innovazione e la collaborazione. Tuttavia, di fronte a questi benefici, dobbiamo fare i conti con un paio di questioni tecniche, le quali costituiscono gli ostacoli principali [23].

1.4.1 Formati Open File

Anche se le informazioni sono fornite in formati elettronici, dettagliati e comprensibili da un computer (*machine-readable*), potrebbero ugualmente verificarsi problemi relativi ai formati del file stesso. I formati in cui le informazioni sono pubblicate, ossia la base digitale in cui le informazioni sono salvate, possono essere sia aperti che chiusi. Un formato aperto è uno in cui le specifiche per il software sono disponibili a tutti e in maniera gratuita, così che chiunque possa utilizzare queste specifiche nel loro software senza limitazioni al riuso imposte dai diritti sulla proprietà intellettuale. D'altro canto, se un formato file è chiuso, potrebbe essere sia perché è proprietario e le specifiche non sono disponibili al pubblico, o perché il formato file è proprietario ma il riuso è limitato, anche se le specifiche sono state rese pubbliche. Se le informazioni sono rilasciate in un formato chiuso, questo può causare ostacoli significativi al riuso delle informazioni in esso racchiuse, costringendo chi sia interessato a utilizzarle a comprare il software necessario. Il beneficio dei formati file aperti è che permettono agli sviluppatori di produrre più pacchetti software e servizi che utilizzano questi formati, minimizzando così gli ostacoli al riuso delle informazioni che contengono. Utilizzare formati file

proprietary per i quali le specifiche non sono disponibili pubblicamente può generare una dipendenza su software di terze parti o sui detentori delle licenze del formato file, il che potrebbe significare, nello scenario peggiore, che sia possibile leggere le informazioni esclusivamente utilizzando certi pacchetti software, i quali potrebbero avere un costo proibitivo, o anche diventare obsoleti. Di conseguenza, per i dati aperti forniti dal governo vige la forte preferenza per cui le informazioni dovrebbero essere rilasciate in formati file aperti e *machine-readable*, leggibili da un computer [22].

1.4.2 Gestione di dataset eterogenei

Il termine *data integration* si riferisce a tutte le azioni necessarie a unificare diverse sorgenti informative, per creare un'unica vista su un determinato processo. Negli scorsi decenni poteva essere considerato normale creare dei silos di dati separati per ogni funzione, ad oggi con i Big Data è possibile configurare un'architettura più moderna.

Integrare quanti più dati possibili è fondamentale per diversi motivi, tra cui i principali sono la ricerca di pattern nascosti e relazioni tra diverse fonti, o la gestione di tipologie di dati che non è possibile immagazzinare con tecnologie tradizionali, come dati semi-strutturati o non-strutturati.

Il valore estraibile dai Big Data non proviene tanto dal volume di dati, quanto dalla correlazione di una varietà di fonti, tipologie e formati di dati. Di conseguenza, la Data Integration è un prerequisito essenziale per sviluppare analisi avanzate, dalle quali estrarre nuova conoscenza.

Tuttavia, la gestione, l'integrazione e la governance di dati eterogenei è una sfida che non viene sempre affrontata in maniera ottimale, a causa della mancanza di tecnologie adeguate e, in alcuni casi, della scarsa consapevolezza del problema.

Le tipologie di dati a disposizione possono essere identificate in:

- Dati machine to machine (M2M): includono quelli generati dall'interazione tra dispositivi elettronici, ad esempio sensori, RFID e connessioni WiFi.

- Dati people to machine: includono i dati generati dall'interazione tra persone e dispositivi elettronici, ad esempio i dati generati durante le transazioni effettuate tramite acquisti online.
- Dati people to people: sono quelli generati dall'interazione tra persone, ad esempio i dati generati sui social network dalle attività degli iscritti, sui forum e blog.
- Public admin data: sono dati presenti in database pubblici e includono gli Open Data, ovvero i dati disponibili gratuitamente a chiunque per essere analizzati e rivisti secondo il proprio interesse, senza restrizioni di copyright, brevetti o altri meccanismi di controllo.
- Enterprise data: sono i dati presenti all'interno dei data warehouse aziendali, ne fanno parte ad esempio i dati degli ERP o del CRM.

L'integrazione dei dati, invece può essere applicata in diversi approcci:

- Silos: rappresenta la modalità tradizionale di immagazzinamento dati, in cui essi vengono raccolti separatamente per ciascuna finalità e utilizzi differenti. Lo storage dei dati è organizzato secondo diversi repositories e gli ambienti sono isolati tra loro, quindi non essendoci comunicazione i dati non sono integrati.
- Data Warehouse (DWH): archivio informatico che raccoglie i dati dai sistemi operazionali aziendali integrandoli con i dati provenienti dalle fonti esterne. Per poter essere gestiti dal DWH, i dati devono essere strutturati, ovvero rappresentati da relazioni descrivibili con tabelle e schemi rigidi.
- Data Lake (DL): è un ambiente di archiviazione dei dati nel loro formato nativo, fino a quando non si rivela necessario dare loro una struttura. Permette l'integrazione di elevate quantità di dati di qualsiasi formato e provenienti da qualsiasi fonte.

- **Modello Integrato:** rappresenta una risposta avanzata alle crescenti esigenze di storage, gestione e analisi dati. Questo approccio offre sia un Data Lake che un Data Warehouse che lavorano in modalità integrata per rispondere alle diverse esigenze di storage, gestione e analisi di qualsiasi tipologia di dato. Il Data Lake offre una vasta capacità di archiviazione per dati non strutturati e di diversa natura, consentendo una raccolta flessibile e scalabile. Il DWN, invece, fornisce una struttura organizzata per l'elaborazione e l'analisi di dati strutturati, garantendo prestazioni ottimali. L'integrazione di entrambe queste componenti consente di affrontare con successo la complessità delle informazioni, rispondendo in modo efficace alle diverse esigenze di gestione e analisi di dati eterogenei. Questo modello si rivela dunque fondamentale nell'era moderna, dove la varietà e la quantità dei dati richiedono approcci dinamici e flessibili per sfruttare appieno il potenziale informativo [24].

1.4.3 Aspetti etici e legali

Quando parliamo di database, dobbiamo innanzitutto effettuare una distinzione tra la struttura e il contenuto: utilizzando il termine *dati* ci riferiamo al contenuto del database, mentre gli elementi strutturali fanno riferimento ad attributi come i nomi dei campi e il modello della struttura dati, ovvero l'organizzazione dei campi sopra citati e le loro interrelazioni.

In molte giurisdizioni è probabile che gli elementi strutturali di un database siano coperti da copyright, anche se ciò dipende in una certa misura dal livello di creatività che è stato coinvolto nella creazione della struttura. Tuttavia, in questo caso siamo particolarmente interessati ai dati. Quando parliamo di dati dobbiamo prestare attenzione, in quanto tale parola non è particolarmente precisa e può riferirsi sia a un insieme che a un singolo elemento, ma può anche significare una grande raccolta di dati, come tutto il contenuto del database.

Per evitare confusione, riserveremo il termine *contenuto* per riferirci ai singoli elementi, mentre la parola *dati* riguarderà l'intera raccolta. Diver-

samente da materiali multimediali come testi, musica o film, la situazione legale per i dati varia notevolmente tra diversi Paesi. Tuttavia, la maggior parte delle giurisdizioni gli garantisce alcuni diritti in qualità di raccolta dati.

La distinzione tra il contenuto del database e l'intera raccolta diviene particolarmente cruciale per i database fattuali, dato che nessuna giurisdizione garantisce il diritto di monopolio sui fatti individuali, ovvero il contenuto, anche se potrebbe garantire loro dei diritti in qualità di raccolta dati.

Per fare un esempio, consideriamo un semplice caso di un database che elenca i punti di fusione di varie sostanze. Mentre l'intero database potrebbe essere protetto dalla legge, così da non permettere a nessuno di accedere, riusare o ridistribuire i suoi dati senza averne il permesso, questo non impedirebbe mai a nessuno di affermare il fatto che la sostanza Y si scioglie alla temperatura Z.

Le forme di protezione per le raccolte dati ricadono largamente in due casi: diritto d'autore e diritto *sui generis*. Come abbiamo già sottolineato, non abbiamo nessuna regola generale e la situazione varia in base alla giurisdizione. Di conseguenza, bisogna procedere considerando singolarmente ciascun Paese per delineare, se applicabile, quale dei due approcci sia utilizzato in una particolare giurisdizione. Dobbiamo inoltre notare come, in assenza di protezione legale, molti fornitori di database chiusi, ovvero non aperti, possono utilizzare un semplice contratto combinato con disposizioni legali per proibire la violazione dei meccanismi di controllo dell'accesso e ottenere così dei risultati simili a quelli dati da un diritto formale sull'indirizzo IP.

Ad esempio, il provider di un database di citazioni può raggiungere qualsiasi insieme di termini e condizioni che voglia implementare, semplicemente richiedendo agli utenti di autenticarsi tramite una password, o fornendo agli utenti un account e una password alla condizione che l'utente accetti i termini e le condizioni del servizio [25].

1.4.4 Opportunità

Sappiamo che gli Open Data hanno le seguenti caratteristiche fondamentali:

- **Accessibilità:** devono essere facilmente reperibili senza restrizioni, così da poter essere utilizzati liberamente.
- **Aggiornamento regolare:** devono essere sempre aggiornati per rimanere attuali e rilevanti nel tempo.
- **Completezza e precisione:** devono essere completi e accurati, garantendo che le informazioni siano attendibili e affidabili.
- **Trasparenza:** devono essere trasparenti, permettendo al cittadino di comprendere al meglio.
- **Sostenibilità:** devono garantire che le risorse necessarie per la pubblicazione e il mantenimento dei dati siano disponibili a lungo termine.

Tra i benefici degli Open Data, invece, ne possiamo trovare svariati:

- **Innovazione e Sviluppo Economico:** forniscono alle imprese l'accesso a informazioni fondamentali per lo sviluppo di nuovi prodotti, servizi e soluzioni, generando nuove opportunità di business e stimolando la crescita economica.
- **Miglioramento dei Servizi Pubblici:** consentono alle autorità locali e ai governi di prendere decisioni più consapevoli e di fornire servizi più efficienti e mirati in settori come la sanità, i trasporti e l'istruzione.
- **Partecipazione Civica:** favoriscono la partecipazione dei cittadini, consentendo loro di accedere, analizzare e utilizzare le informazioni per influenzare il processo decisionale e promuovere il cambiamento sociale positivo.

- Trasferimento di Conoscenze e Capacità: facilitano il trasferimento di conoscenze e capacità tra individui, organizzazioni e Paesi, consentendo la collaborazione e lo scambio a livello locale, nazionale e internazionale.
- Miglioramento della Qualità della Vita: forniscono ai cittadini l'accesso a informazioni cruciali su questioni come salute, ambiente e sicurezza pubblica, insieme alla relativa qualità dei servizi.
- Ricerca e Innovazione Scientifiche: forniscono agli scienziati e ai ricercatori l'accesso a dati di alta qualità che possono essere utilizzati per condurre studi e sviluppare soluzioni a sfide globali come il cambiamento climatico, le malattie e le crisi umanitarie [23].

Tecnologie Utilizzate

In questo capitolo presenteremo le varie tecnologie utilizzate per la realizzazione del progetto, descrivendole in modo da fornire una conoscenza di base abbastanza solida da permettere la comprensione dell'ambiente.

2.1 HTML

HTML, acronimo che significa *HyperText Markup Language*, è un linguaggio di markup che permette di impaginare e formattare pagine web collegate tra loro tramite link. Un ipertesto non è altro che l'albero di navigazione che collega le pagine web, ovvero un flusso infinito di pagine collegate tra loro attraverso dei link che permettono di spostarsi da un contenuto all'altro.

HTML risponde all'esigenza di riuscire a pubblicare del testo online, mantenendo la formattazione e il significato di ciascuna delle sue parti, utilizzando dei marcatori detti `<tag>`. Il browser legge il tag e il suo contenuto, quindi traduce a schermo il codice usando i criteri specificati da HTML. Questo ha permesso di costruire pagine con una struttura simile fra di loro e soprattutto replicabile seguendo uno standard. Nel giro di pochi anni furono aggiunti via via sempre più tag ed elementi per consentire la creazione di pagine contenenti immagini, elementi interattivi, form, pulsanti, tabelle e altri ancora. Pertanto, HTML si è trasformato in un linguaggio molto completo ma in continuo mutamento, che oggi viene mantenuto dal World Wide Web Consortium (W3C), una associazione non governativa che si oc-

cupa di implementare nuove funzioni per rendere il web sempre più libero e accessibile.

Riguardo ai tag, sono alla base dell'HTML e ciascuno di essi corrisponde a un determinato tipo di contenuto. Ogni tag può avere degli attributi specifici, cosa che permette di costruire pagine diverse tra di loro e in modo tale che rispondano alle necessità di chi le scrive. Le pagine in HTML hanno una struttura ad albero: proseguendo lungo la ramificazione, si possono trovare più o meno elementi che costruiscono la pagina stessa seguendo una precisa gerarchia. Ad esempio, nel seguente frammento di codice:

```
<html>
  <head>
    <title>Titolo</title>
  </head>
  <body>
    <p>Paragrafo</p>
  </body>
</html>
```

`<html>` indica l'inizio della parte di codice che verrà espressa utilizzando il linguaggio HTML. Tranne alcuni tag detti *self-closing tags*, tutti vanno chiusi mediante il rispettivo tag di chiusura, che in questo caso è `</html>`. `<head>` specifica l'header della pagina, il quale racchiude delle informazioni importanti per il suo funzionamento, ma invisibili dal nostro dispositivo. Contiene a sua volta un `<title>`, ovvero un titolo che è quello rappresentativo della pagina stessa e del suo contenuto. Una volta chiusa l'intestazione, si passa al `<body>`, cioè il contenuto della pagina. Al suo interno troviamo il tag `<p>`, utilizzato per scrivere paragrafi di testo, al cui interno viene scritto il rispettivo contenuto. Tutti i vari browser web sono pensati per interpretare i tag più o meno ugualmente ogni volta, ma ci possono essere eccezioni in quanto ogni browser implementa un proprio rendering della pagina web.

HTML non è un linguaggio di programmazione, bensì un linguaggio di markup: descrive al browser com'è fatta la struttura di una pagina, e niente

più. Un linguaggio di programmazione, invece, ha un ruolo funzionale: risolve cicli di codice seguendo una struttura fatta di `if` e `else`, può svolgere calcoli matematici, può manipolare dati e variabili. Quindi, è il browser web che è programmato per capire la struttura delle pagine scritte in HTML, mentre quest'ultimo descrive soltanto la struttura della pagina e del suo contenuto. Di conseguenza, HTML non è un linguaggio di programmazione, mentre lo sono ad esempio JavaScript e PHP.

Ad oggi non è più sufficiente utilizzare solo HTML per realizzare contenuti web, in quanto le esigenze sono da tempo cambiate: ai siti web serve più di soli contenuti testuali, quindi vengono usati anche altri linguaggi, tra cui CSS e JavaScript, che consentono di scrivere pagine ricche e complete, sia nell'aspetto di grafica che nel comportamento [26].

2.2 CSS

CSS, acronimo di *Cascading Style Sheets*, è un linguaggio di stile per i documenti web. I CSS forniscono istruzioni a un browser o a un altro programma utente su come il documento debba essere presentato all'utente, definendone proprietà come il font, i colori, le immagini di sfondo, il layout, il posizionamento delle colonne o di altri elementi della pagina.

CSS è un linguaggio nato per essere il complemento ideale di HTML, per questo i due linguaggi hanno sempre proceduto parallelamente: nelle intenzioni del W3C, HTML è un semplice linguaggio strutturale, estraneo a qualunque scopo che riguardi la presentazione di un documento. Invece, CSS è lo strumento designato proprio per arricchire l'aspetto visuale e la presentazione di una pagina, nato per separare il contenuto dalla presentazione [27].

Anche CSS è un *living standard* come HTML e per questo viene regolarmente sviluppato dal W3C. Il suo compito principale è definire il design di un sito web e per questo scopo vengono assegnati determinati valori agli elementi HTML con l'aiuto delle proprietà del linguaggio di sti-

le. Queste regole hanno una struttura di base che corrisponde allo schema `selettore { dichiarazione }`. Un selettore non è altro che una rappresentazione dell'elemento HTML al quale fa riferimento la regola, mentre la dichiarazione consiste dalla combinazione tra proprietà e valore che viene annotata tra due parentesi graffe. Ogni dichiarazione finisce con un punto e virgola, come ad esempio `h2 { color: #ff0000; }`, nel quale il selettore `h2` rappresenta le intestazioni di secondo ordine, ovvero i sottotitoli. La dichiarazione, invece, utilizza la proprietà `color` per colorare tali sottotitoli di rosso, in questo caso utilizzando un colore espresso in esadecimale per definire impostazioni di colore precise.

CSS supporta diversi selettori in grado di dare istruzioni di regole:

- **selettore** corrisponde al nome dell'elemento HTML a cui fa riferimento: lo stile viene applicato su tutti gli elementi HTML dello stesso tipo.
- **.selettore** si rivolge a tutti gli elementi di una classe specifica e si scrive con un punto davanti al nome della classe HTML corrispondente.
- **#selettore** si riferisce a un unico elemento con un ID specifico. L'integrazione nel codice sorgente HTML avviene grazie all'attributo `id`, ovvero `id="identificatore"`.
- ***** è il selettore universale e si rivolge a tutti gli elementi HTML di un documento.

Il CSS può essere dichiarato in vari modi nell'HTML:

- **Inline**: viene definito direttamente sul tag dell'elemento HTML utilizzando l'attributo `style`. Ha il vantaggio che non è necessario configurare un foglio di stile apposito e ha la priorità elevata, ma nel momento in cui deve essere applicato un blocco di dichiarazioni questo metodo di stile diventa poco chiaro e ridondante.
- **Interno**: viene definito all'interno del tag `<style>` nella `<head>` del documento HTML.

- Esterno: viene definito in un foglio di stile separato, il quale verrà collegato al documento HTML di base utilizzando l'elemento HTML `<link>`, sempre nella `<head>`. Può essere riutilizzato in più pagine HTML, semplicemente importandolo ove necessario.

2.3 SCSS

SCSS, acronimo di *Sassy Cascading Style Sheets*, è un'estensione di CSS molto popolare che aggiunge funzionalità potenti che aumentano le capacità standard di CSS.

In qualità di linguaggio di script preprocessing, SCSS permette agli sviluppatori di utilizzare variabili, regole annidate (*nested*), *mixins* e funzioni, che semplificano il processo di scrittura e mantenimento di stylesheets complessi. Essendo un superset di CSS, SCSS è pienamente compatibile con CSS e rende più facile creare codice pulito, organizzato e riutilizzabile, migliorando l'efficienza e la scalabilità dei progetti di web development.

SCSS è una delle due possibili sintassi per il preprocessore SASS, o *Syntactically Awesome Style Sheets*. Come ogni preprocessore, SASS viene compilato in codice CSS nativo che funziona su ogni web browser. La vera potenzialità di questo linguaggio si vede sul lato sviluppatore, in quanto permette di scrivere codice SCSS conciso che verrà compilato in un codice CSS più lungo. Per questo, gli sviluppatori possono riuscire a fare di più scrivendo meno, senza compromettere la compatibilità col web.

La differenza principale tra SCSS e CSS è che SCSS è una sintassi per il preprocessore SASS, mentre CSS è un linguaggio di stile che descrive come un browser debba visualizzare gli elementi HTML. SCSS supporta da molto tempo l'utilizzo di variabili in qualsiasi punto dello stylesheet, mentre CSS offre un supporto nativo per le variabili che è relativamente recente e può essere usato solo per salvare valori e tokens della UI. In SCSS, la sintassi può essere annidata, permettendo di effettuare il nesting sia delle proprietà che di altri selettori, mentre in CSS un singolo selettore può annidare proprietà ma

non altri selettori. Inoltre, SCSS supporta i mixins, che sono gruppi di dichiarazioni CSS che possono essere riutilizzate all'interno dello stylesheet, il che può essere utile quando utilizziamo prefissi vendor (come `-webkit-` per Chrome o `-moz-` per Firefox), animazioni complesse, o altro codice che potrebbe essere riutilizzato in più posti.

La sintassi SCSS e CSS hanno delle somiglianze in comune, ma quella SCSS permette l'utilizzo di funzionalità più avanzate. SCSS utilizza i punti e virgola e l'indentazione per mantenere la formattazione, mentre il normale CSS potrebbe essere più flessibile senza tuttavia offrire le funzionalità avanzate di SCSS. L'utilizzo di SCSS permette di mantenere una codebase più pulita e più facilmente mantenibile.

Per compilare SCSS in CSS si utilizza un compiler SASS, che funziona con entrambe le sintassi SASS e SCSS [28]. Queste due sintassi sono equivalenti, l'unica differenza è che in SASS si elimina l'utilizzo di punti e virgola e parentesi graffe mediante una sintassi *whitespace-sensitive*, rendendolo però incompatibile con CSS [29].

2.4 JavaScript

JavaScript è un linguaggio di scripting, o di programmazione, che permette di implementare funzionalità complesse all'interno delle pagine web, ad esempio ogni volta che un sito debba essere dinamico e non statico, per creare contenuti aggiornati automaticamente, mappe interattive, controllare elementi multimediali, oppure grafiche animate in 2D o 3D. Costituisce il terzo strato dello stack delle tecnologie web standard.

Lato client, il linguaggio JavaScript consiste delle solite funzionalità di programmazione come il salvataggio di valori in variabili, le operazioni su stringhe, e l'esecuzione di codice in risposta a certi eventi che avvengono su una pagina web, come quello di `click`. Ma ancora più importante, ci permette di utilizzare le funzionalità costruite su altro codice JavaScript client-side, mediante le *Application Programming Interfaces*, o APIs. Queste sono bloc-

chi di codice già pronto che consentono a uno sviluppatore di implementare programmi che altrimenti sarebbero difficili o impossibili da implementare, rendendo questi compiti più semplici da realizzare. Le API possono essere built-in nel browser o di terze parti. Queste ultime verranno approfondite in una sezione successiva. Le API del browser permettono di esporre dati dall'ambiente del computer, o effettuare compiti complessi. Per esempio:

- DOM (*Document Object Model*) API: permettono di manipolare HTML e CSS, creando, rimuovendo e cambiando l'HTML, applicando nuovi stili dinamicamente alla pagina, ecc. Per esempio, vengono utilizzate ogni volta che vediamo aprirsi una finestra popup su una pagina, o quando viene visualizzato del nuovo contenuto.
- Geolocation API: permettono di recuperare informazioni geografiche, come la posizione GPS corrente.
- Canvas e WebGL API: permettono di creare animazioni 2D e 3D.
- Audio e Video API: permettono di riprodurre audio e video direttamente in una pagina web, o filmare un video dalla webcam e visualizzarlo su un altro computer.

JavaScript è un linguaggio leggermente interpretato: il browser riceve il codice JS originale e lancia lo script, e nei browser moderni viene utilizzata una tecnica di compilazione detta *just-in-time* per migliorare le performance. In questo caso, il codice JS viene compilato in un formato binario più veloce mentre lo script è in esecuzione, così da lanciarlo il più rapidamente possibile. Tuttavia, JavaScript è comunque considerato un linguaggio interpretato, dato che la compilazione avviene a *run-time*, invece che *ahead-of-time*.

JavaScript può essere utilizzato sia client-side che server-side. Lato client, viene eseguito sul computer dell'utente, ad esempio quando visualizziamo la pagina web, viene scaricato il codice client-side del sito web, quindi è eseguito e visualizzato dal browser. Lato server, invece, viene eseguito dal server, per poi inviare i risultati al browser lato client, dove verranno scaricati

e visualizzati. Tra i vari linguaggi server, possiamo utilizzare JavaScript nell'ambiente Node.js.

Inoltre, è utilizzato per creare pagine web dinamiche, dove *dinamico* si riferisce all'abilità, sia in client- che server-side JS, di aggiornare la visualizzazione di una webpage o webapp per mostrare cose diverse in circostanze diverse, generando il nuovo contenuto richiesto. Il lato server genera dinamicamente il nuovo contenuto, prendendo i dati dal database, mentre lato client JavaScript crea dinamicamente nuovo contenuto all'interno del browser, prendendo i dati che gli sono stati inviati dal backend. Generalmente questi due approcci lavorano insieme. Una pagina web statica, invece, non aggiorna il contenuto dinamicamente e mostra sempre la stessa schermata [30].

2.4.1 AJAX

AJAX, acronimo di *Asynchronous JavaScript and XML*, è una tecnica di sviluppo web in cui una webapp scarica contenuto dal server mediante una richiesta HTTP asincrona, quindi lo utilizza per aggiornare le parti della pagina rilevanti, senza bisogno di ricaricare nuovamente la pagina. Questo può rendere la pagina più responsive, in quanto si richiedono solo le parti che necessitano di essere aggiornate.

AJAX può essere utilizzato per creare *single-page apps*, dove l'intera webapp consiste in un singolo documento che usa AJAX per aggiornare il contenuto in base alle richieste.

Inizialmente, AJAX venne implementato utilizzando l'interfaccia `XMLHttpRequest`, ma l'API di `fetch()` è più adatta alle applicazioni web moderne, in quanto è più potente, flessibile, e in grado di integrarsi meglio con le tecnologie web fondamentali come i *service workers*. I framework web moderni offrono astrazioni per AJAX. Tuttavia, ad oggi questa tecnica è talmente diffusa che il termine specifico *AJAX* viene utilizzato raramente [31].

2.4.2 Framework e librerie

In JavaScript sono disponibili un gran numero di framework e librerie, sviluppati nel corso degli anni per velocizzare e semplificare lo sviluppo web.

Le librerie sono dei pacchetti di classi e funzioni riutilizzabili per risolvere particolari problemi, sono flessibili e possono essere utilizzate liberamente. Alcuni esempi di librerie famose sono:

- React.js, utilizzato per creare UI complesse e gestire lo stato.
- jQuery, utilizzato per gestire eventi JS ed effettuare chiamate AJAX.
- D3.js (*Data-Driven Documents*), utilizzato per la visualizzazione dati.
- Chart.js, utilizzato per disegnare vari tipi di grafici utilizzando l'elemento HTML5 `<canvas>`.

I framework, invece, offrono un approccio strutturato allo sviluppo di un progetto, mediante un insieme di strumenti utili a creare applicazioni in maniera standardizzata. Specificano dove inserire quali frammenti di codice e spesso includono funzionalità come la gestione del routing, richieste HTTP, o dei dati di backend. Questo approccio strutturato semplifica lo sviluppo e assicura consistenza tra i vari progetti. Alcuni esempi di framework famosi sono:

- Angular, sviluppato da Google, utilizzato per lo sviluppo cross-platform.
- Vue.js, utilizzato per creare UI e SPAs (*Single-Page Applications*).
- Next.js, framework basato su React per sviluppare webapp multiplatforma [32].

È possibile anche eseguire codice JavaScript al di fuori dal browser, utilizzando l'ambiente di esecuzione Node.js, che è open-source e multiplatforma [33].

2.5 Leaflet

Leaflet è una libreria JavaScript open-source che permette di visualizzare mappe interattive mobile-friendly. Il suo codice pesa solo 42KB, ma contiene tutte le funzionalità di mapping che possono servire a quasi tutti gli sviluppatori.

Questa libreria è stata progettata avendo in mente la semplicità, le prestazioni e l'usabilità. Funziona in modo efficiente su tutte le principali piattaforme desktop e mobile, permette l'estensione delle sue funzionalità mediante l'utilizzo di plugins e offre diverse API.

Leaflet non cerca di fare tutto, ma si focalizza sulle funzionalità di base per fare in modo che funzionino perfettamente. Le sue caratteristiche principali sono:

- **Layers**
Offre i cosiddetti *tile layers*, i *markers*, e i popup. Inoltre, si possono mettere delle immagini come overlays e i GeoJSON. Sulla mappa si possono disegnare poligoni, polylines e cerchi.
- **Interazione**
Supporta lo scorrimento, lo zoom tramite gesture, doppio click o scroll wheel, lo spostamento dei marker, gli eventi, e altro ancora.
- **Funzionalità**
Visivamente, supporta le animazioni per lo zoom e il fade. I popup possono essere customizzati tramite CSS, e si può creare un marker personalizzato tramite immagini o HTML. Riguardo alle mappe, gli sviluppatori possono utilizzare i layer che preferiscono.
- **Controlli**
Sono presenti bottoni per lo zoom e il cambio di layer, un'indicatore per la scala, e uno per l'attribuzione [34].

2.6 Vue.js

Vue è un framework JavaScript utilizzato per la costruzione di interfacce utente (UI). Si basa sullo standard stack HTML, CSS e JavaScript e offre un modello di programmazione dichiarativo e basato sui componenti, che aiuta a sviluppare in modo efficiente delle UI anche complesse. Le funzionalità chiave di Vue sono due:

- **Rendering dichiarativo:** Vue estende lo standard HTML mediante una sintassi di template che permettono di descrivere output HTML in modo dichiarativo, basato su uno stato definito in JavaScript.
- **Reattività:** Vue traccia automaticamente i cambiamenti di stato in JavaScript e aggiorna il DOM in modo efficiente quando avvengono queste variazioni.

Vue è un framework ed ecosistema che copre la maggioranza delle più comuni funzionalità richieste nello sviluppo frontend. Tuttavia, il web è estremamente vario e ciò che viene costruito può cambiare drasticamente in forma e scala. Vue è stato disegnato proprio per rispondere a queste esigenze, in modo da essere flessibile e adottabile in modo incrementale. In base al caso d'uso, Vue può essere utilizzato in diverse modalità:

- Miglioramento di HTML statico senza necessità di compilazione.
- Integrazione di componenti web su qualsiasi pagina.
- Creazione di *Single-Page Applications* (SPA).
- Utilizzo di Fullstack o *Server-Side Rendering* (SSR).
- Utilizzo di Jamstack o *Static Site Generation* (SSG).
- Targeting specifico di desktop, mobile, WebGL, o anche il terminale.

Vue è flessibile, ma le sue nozioni principali sono condivise a tutti i livelli, rendendolo un framework adattabile alle esigenze di chiunque [35].

2.7 NPM

NPM, o *Node Package Manager*, è il package manager di default per Node.js. Nel settembre 2022, il registro di NPM conteneva oltre 2.1 milioni di pacchetti, rendendolo il più grande repository al mondo di codice che utilizza un solo linguaggio, possiamo quindi essere quasi sicuri che esista un pacchetto per ogni evenienza. NPM nacque come un modo per scaricare e gestire le dipendenze di Node.js, ma è in seguito diventato uno strumento utilizzato anche nel JavaScript a frontend.

NPM installa, aggiorna e gestisce i download delle dipendenze del progetto. Le dipendenze sono blocchi di codice precostruiti, come le librerie e i pacchetti, di cui l'applicazione ha bisogno per poter funzionare.

Oltre ai normali download, NPM gestisce anche il versioning, così da poter specificare qualsiasi versione specifica di un pacchetto, o richiedere una versione più recente o più vecchia di quella che ci serve. Spesso capita che una libreria sia compatibile solamente con una major release di un'altra libreria, o che un bug nell'ultima versione di una libreria sia ancora irrisolto e continui a causare problemi. Specificare esplicitamente una versione di una libreria aiuta anche a tenere tutto il team sulla stessa versione di un package, così che tutti la utilizzino fino a quando il file `package.json` sia aggiornato. In tutti questi casi, il controllo delle versioni aiuta notevolmente e NPM segue lo stesso standard di *semantic versioning*, o *semver*.

NPM consente anche agli sviluppatori di eseguire comandi personalizzati, supportando un formato per specificare task a terminale che vengono lanciati come scripts. Per esempio, utilizzando `npm run dev` possiamo lanciare l'app in modalità debug senza scrivere l'effettivo comando corrispondente, che sarebbe molto più lungo. Altri comandi, tra tutti, sono `npm run prod` e `npm run build` [36].

2.8 PHP

PHP, acronimo ricorsivo per *PHP: Hypertext Preprocessor*, è un linguaggio di scripting open-source e general-purpose molto usato, che è specialmente adatto per lo sviluppo web e ad essere integrato nell'HTML.

Invece di utilizzare molti comandi per visualizzare l'HTML, come avviene in C o Perl, le pagine PHP contengono HTML all'interno di codice integrato che svolge qualche preciso compito. Il codice PHP è racchiuso in speciali istruzioni di inizio e fine, ovvero `<?php` e `?>` che permettono di entrare e uscire dalla *modalità PHP*.

Ciò che distingue PHP dal codice JavaScript client-side è che il codice è eseguito sul server, generando HTML che viene quindi inviato al client. Il client riceve così il risultato dell'esecuzione di quello script, senza sapere quale sia il codice sottostante. Un web server può essere configurato per processare tutti i file HTML utilizzando PHP, senza che gli utenti abbiano alcun modo per capire che si stia utilizzando PHP.

PHP riguarda principalmente lo scripting lato server, quindi può fare qualsiasi cosa che altri programmi CGI (*Common Gateway Interface*) sanno fare, come raccogliere dati da un *form*, generare dinamicamente il contenuto della pagina, o inviare e ricevere cookies. Due sono le aree principali dove gli script PHP vengono utilizzati:

- Scripting lato server

Questo è il più utilizzato e costituisce il principale campo di applicazione per PHP. Per funzionare, richiede un parser PHP (CGI o modulo server), un server web, e un browser web. Tutti questi possono essere eseguiti localmente per scopi di sviluppo.

- Scripting a command line

Uno script PHP può essere eseguito senza richiedere alcun server o browser, basta solo disporre di un parser PHP per poterlo utilizzare in questa maniera. Questo tipo di utilizzo è ideale per script eseguiti

regolarmente tramite **cron**, su Unix e macOS, o con *Task Scheduler* su Windows. Questi script possono essere semplicemente usati anche per processare del testo.

PHP può essere utilizzato su tutti i principali sistemi operativi, tra cui Linux, molte varianti Unix come Solaris e OpenBSD, Microsoft Windows, macOS, RISC OS, e altri ancora. PHP supporta anche la maggior parte degli attuali web server, inclusi Apache, IIS, e molti altri, incluso qualsiasi web server in grado di utilizzare *binaries* di FastCGI, come *lighttpd* e *nginx*. PHP è in grado di lavorare come modulo o come processore CGI.

Con PHP, gli sviluppatori hanno la libertà di scelta sia per il sistema operativo che per il web server. Inoltre, viene data la possibilità di scegliere se utilizzare un paradigma di programmazione procedurale o uno ad oggetti (OOP), o addirittura di mescolarli.

PHP non si limita a sfornare HTML, ma è in grado di generare file RTF (*Rich File Types*), come immagini o file PDF, criptare dati, e inviare emails. Inoltre, può produrre facilmente qualsiasi tipologia di testo, come JSON o XML. PHP è in grado di generare questi file automaticamente e di salvarli a file system, invece di mostrarli a schermo, formando così una cache a lato server per il contenuto dinamico.

Una delle funzionalità più importanti di PHP è il suo supporto a una grande varietà di database. Scrivere una pagina web che acceda a un db diventa così incredibilmente semplice, utilizzando una delle estensioni specifiche come *mysqli* per MySQL, o mediante un livello di astrazione come PDO (*Protected Destination of Origin*), oppure connettendosi a un database che supporta lo standard Open Database Connection tramite l'estensione ODBC. Altri database, invece, possono utilizzare *cURL* o *sockets*.

PHP supporta molti protocolli per la comunicazione con altri servizi, ad esempio IMAP, POP3 e HTTP, oltre a molti altri, ed è in grado di aprire dei socket grezzi di rete e interagirvi con qualsiasi altro protocollo. PHP supporta lo scambio di dati complessi tra virtualmente tutti i linguaggi di programmazione utilizzati sul Web. Riguardo all'interconnessione, PHP offre

supporto per istanziare oggetti Java e utilizzarli in modo trasparente come oggetti PHP [37].

2.9 API

Le APIs sono meccanismi che permettono a due componenti software di comunicare tra loro utilizzando un insieme di definizioni e protocolli. Ad esempio, vengono utilizzate dalle app meteo nello smartphone, le quali scaricano i dati meteorologici giornalieri effettuando una richiesta alle API meteo corrispondenti per poter visualizzare gli aggiornamenti.

API significa *Application Programming Interface*. Nel contesto delle APIs, Applicazione si riferisce a ogni software con una funzione distinta, mentre Interfaccia è un contratto di servizio tra due applicazioni, che definisce come queste due possano comunicare tra loro utilizzando richieste e risposte. La loro documentazione API contiene informazioni su come gli sviluppatori debbano strutturare tali richieste e risposte.

L'architettura API è generalmente spiegata in termini di client e server. L'applicazione che invia la richiesta è detta client, mentre quella che manda la risposta è il server. Quindi, nell'esempio di prima, il server è il database meteo del provider, mentre l'app mobile costituisce il client.

2.9.1 Tipologie di API

Esistono quattro diversi modi in cui le APIs possono funzionare, in base a quando e perché sono state create.

- SOAP APIs

Queste API utilizzano un protocollo chiamato *Simple Object Access Protocol*, dove client e server si scambiano messaggi tramite XML. Questa tipologia di API è meno flessibile, ma è stata più popolare in passato.

- RPC APIs

Queste API sono chiamate *Remote Procedure Calls*: il client completa una funzione (o procedura) sul server, quindi il server invia il corrispondente output al client.

- WebSocket APIs

Sono un'altra tipologia di API utilizzate nel moderno sviluppo web e utilizzano oggetti JSON per il passaggio dei dati. Un'API di tipo WebSocket supporta la comunicazione a due vie tra le app client e il server. Il server, a sua volta, può inviare messaggi di callback ai client connessi, rendendolo così più efficiente delle API di tipo REST.

- REST APIs

Sono le API più popolari e flessibili che possiamo trovare, ad oggi, sul web. Il client invia richieste al server sotto forma di dati, quindi il server utilizza questo input fornito dal client per eseguire funzioni interne e ritorna i dati di output, inviandoli indietro al client.

Oltre all'architettura, le API sono classificate anche in base al loro scopo di utilizzo:

- API Private

Sono interne a un'azienda e vengono utilizzate solamente per connettere sistemi e dati all'interno di essa.

- API Pubbliche

Sono aperte al pubblico e possono essere utilizzate da chiunque. Tuttavia, potrebbe esserci qualche tipo di autorizzazione e costo associato al loro utilizzo, anche se non necessariamente.

- API Partner

Sono accessibili solo agli sviluppatori esterni autorizzati, per incoraggiare le partnership tra business (B2B).

- API Composite

Combinano due o più APIs per risolvere requisiti di sistema o comportamenti complessi.

2.9.2 RESTful APIs

REST significa *Representational State Transfer* e definisce un insieme di funzioni come GET, PUT, DELETE, e altre, che i client possono utilizzare per accedere ai dati del server. Sia client che server si scambiano dati mediante HTTP.

La funzionalità principale delle API di rest è la mancanza di stato (*statelessness*), ovvero i server non salvano i dati del client tra le varie richieste. Le richieste che il client effettua al server sono simili agli URL utilizzati per visitare un sito web in un normale browser. La risposta fornita dal server consiste in *plain data*, senza alcun tipo di rendering grafico, a differenza di una pagina web.

Una Web API, detta anche Web Service API, è una *application programming interface* tra un web server e un browser. Tutti i web services sono API, ma non tutte le API sono servizi web. Le API REST sono un tipo speciale di Web API che utilizza lo stile architetturale standard che abbiamo visto poco fa.

I diversi termini riguardanti le APIs, come Java API o service API, esistono perché storicamente le API furono create prima del World-Wide Web. Le API web moderne sono REST APIs e i termini possono ormai essere utilizzati in modo intercambiabile.

2.9.3 Utilizzo delle API

Le integrazioni delle API sono componenti software che aggiornano automaticamente i dati tra client e server. Alcuni esempi di integrazioni API sono quando la galleria del telefono viene sincronizzata automaticamente sul cloud, o la data e l'ora del portatile si sincronizzano automaticamente,

aggiornandosi quando viaggiamo in un altro fuso orario. Possono inoltre essere utilizzate dalle aziende per automatizzare molte funzioni di sistema in maniera efficiente.

Le REST APIs offrono quattro principali benefici:

1. Integrazione

Le API sono utilizzate per integrare nuove applicazioni con sistemi software esistenti. Questo accelera la velocità di sviluppo, in quanto ciascuna funzionalità non ha bisogno di essere riscritta da zero, ma possiamo utilizzare le API per sfruttare codice già esistente.

2. Innovazione

L'arrivo di una nuova app è in grado di cambiare intere industrie, quindi le aziende devono essere in grado di rispondere velocemente e supportare il rilascio rapido di servizi innovativi. Questo può avvenire apportando cambiamenti a livello delle API, senza dover riscrivere l'intero codice.

3. Espansione

Le API presentano un'opportunità unica per le aziende di soddisfare le esigenze dei clienti su diverse piattaforme. Per esempio, le Maps API permettono l'integrazione di informazioni relative alle mappe su siti web, Android, iOS, e altri. Qualunque azienda può offrire un simile accesso ai propri database interni, utilizzando API gratuite o a pagamento.

4. Facilità di manutenzione

Una API funge da porta di passaggio tra due sistemi. Ciascun sistema è obbligato ad apportare cambiamenti interni così da non influenzare il funzionamento delle API. In questo modo, ogni cambiamento futuro al codice effettuato da una qualunque delle due parti non impatterà l'altra.

Gli API endpoints sono il punto finale del sistema di comunicazione delle API e includono URL di server, servizi, e altre locazioni specifiche digitali da

cui vengono inviate e ricevute le informazioni tra i diversi sistemi. Gli API endpoints sono di importanza critica per le aziende per due ragioni principali:

1. Sicurezza

Gli API endpoints rendono il sistema vulnerabile agli attacchi, quindi il monitoraggio delle API è cruciale per prevenire un utilizzo improprio.

2. Prestazioni

Gli API endpoints, specialmente quelli ad alto traffico, possono causare bottlenecks (colli di bottiglia) e influenzare le performance del sistema.

Tutte le API quindi devono essere rese sicure tramite appropriata autenticazione e monitoraggio. I due modi principali per mettere in sicurezza le REST APIs includono:

1. Token di autenticazione: sono utilizzati per autorizzare gli utenti a effettuare le chiamate alle API, controllano che gli utenti siano correttamente identificati e che abbiano effettivi diritti di accesso per quella particolare chiamata API. Per esempio, quando effettuiamo l'accesso al server email, il client email utilizza token di autenticazione per un accesso sicuro.
2. Chiavi API: verificano il programma o l'applicazione che effettua la chiamata alle API. Identificano l'applicazione e assicurano che abbia i diritti di accesso richiesti per effettuare quella particolare chiamata API. Le chiavi API non sono sicure come i tokens, ma permettono di monitorare le API per raccogliere dati sul loro utilizzo. Ad esempio, quando visitiamo un sito web e notiamo una lunga stringa di caratteri e numeri nel suo URL, questa stringa è una chiave API che il sito utilizza per effettuare chiamate alle API interne.

2.9.4 Documentazione

Scrivere una documentazione delle API comprensiva è parte del processo di gestione delle API. Tale documentazione può essere generata automatica-

mente tramite appositi strumenti o scritta manualmente. Come *best practice*, è buona norma scrivere le spiegazioni in linguaggio semplice e facile da leggere, preferibilmente in inglese. I documenti generati con tools possono diventare verbosi e quindi richiedere modifiche manuali. Inoltre, bisogna utilizzare esempi di codice per spiegare le funzionalità e mantenere la documentazione accurata e aggiornata. Lo stile di scrittura va rivolto ai principianti e bisogna coprire tutte le tipologie di problemi che le API possono risolvere per gli utenti [38].

2.9.5 Evoluzione delle API

I framework e le librerie possono cambiare le loro APIs. Migrare un'applicazione alle nuove API è tedioso e distrugge il processo di sviluppo: anche se sono stati proposti alcuni strumenti e idee per risolvere l'evoluzione delle API, la maggioranza degli aggiornamenti viene ancora fatta manualmente. Generalmente, i cambiamenti che causano la rottura delle applicazioni esistenti non sono casuali, ma ricadono in particolari categorie: oltre l'80% di questi è dovuto a *refactorings*, il che suggerisce che per l'aggiornamento delle applicazioni dovrebbero essere utilizzati dei tool di migrazione basati sul refactoring [39].

2.10 JSON

JavaScript Object Notation, o JSON, è un formato standard text-based per rappresentare dati strutturati basati su sintassi JavaScript ad oggetti. Viene comunemente utilizzato per trasmettere dati in applicazioni web, ad esempio per inviare dati dal server al client, così da poterlo visualizzare su una pagina web, o viceversa.

Anche se JSON ricorda molto la sintassi JavaScript per creare gli oggetti letterali, può essere utilizzato indipendentemente da JS e molti ambienti di programmazione offrono la capacità di leggere (parse) e generare JSON.

JSON esiste sotto forma di stringa, utile quando vogliamo trasmettere dati sulla rete. Quando vogliamo accedere ai dati, bisogna convertirlo in un oggetto JavaScript nativo, ma questo non è un problema in quanto JS offre un oggetto JSON globale che dispone di metodi per la conversione tra i due. Convertire una stringa in un oggetto nativo è detto *deserializzazione*, mentre il processo inverso in cui si converte un oggetto nativo in una stringa è detto *serializzazione*. Una stringa JSON può essere salvata in un suo proprio file, che consiste praticamente in un file di testo avente l'estensione `.json` e un tipo MIME di `application/json`.

All'interno di un oggetto JSON, possiamo includere gli stessi tipi di dati basilari che utilizzeremo in un oggetto JavaScript standard, ovvero stringhe, numeri, array, booleani e altri ancora. Questo ci permette di costruire una gerarchia di dati che può essere anche annidata e permette l'accesso nello stesso modo in cui si accede ai campi di un analogo oggetto [40].

2.11 SQL

Structured Query Language (SQL) è un linguaggio di programmazione utilizzato per salvare e processare informazioni all'interno di un database relazionale, ovvero in forma tabellare, con righe e colonne che rappresentano diversi attributi e le varie relazioni tra i valori. Possiamo utilizzare le istruzioni SQL per salvare, aggiornare, rimuovere, cercare e raccogliere informazioni dal database, inoltre SQL permette di mantenere e ottimizzare le prestazioni del database [41].

2.11.1 MySQL

MySQL è un *Relational Database Management System* (RDBMS) open-source molto popolare, utilizzato per salvare e gestire dati in maniera affidabile, prestante, scalabile e facile da usare. Da qui in poi ci riferiremo a MySQL come versione del database SQL.

2.11.2 Struttura dei dati relazionali

MySQL è un database relazionale open source che utilizza SQL per creare e gestire i database, salvando i dati in tabelle di righe e colonne organizzate in schemi. Uno schema definisce come i dati sono organizzati e salvati, descrivendo le relazioni esistenti tra le varie tabelle. Con questo formato, possiamo facilmente salvare, raccogliere e analizzare diversi tipi di dati, inclusi semplice testo, numeri, date, orari e, recentemente, JSON e arrays.

Due capacità principali di MySQL sono il suo supporto alle transazioni ACID e la sua abilità di essere scalabile. ACID significa *Atomicity, Consistency, Isolation, and Durability*, ovvero le quattro proprietà che assicurano che le transazioni nel database siano processate in modo affidabile e accurato. Mediante le transazioni ACID, MySQL garantisce che tutte le modifiche ai dati siano effettuate in maniera coerente e affidabile, anche nel caso in cui avvenga un guasto nel sistema. MySQL è in grado di scalare per supportare database di dimensioni molto grandi, gestendo un alto volume di connessioni concorrenti. Le prestazioni, la facilità d'uso e il costo contenuto, insieme alla sua capacità di scalare affidabilmente, hanno reso MySQL il database open source più popolare al mondo.

MySQL è veloce, affidabile, scalabile e facile da utilizzare. Originariamente venne sviluppato per gestire rapidamente grandi database ed è stato utilizzato da molti anni in ambienti di produzione con requisiti molto elevati. MySQL offre un ampio insieme di funzioni di utilità ed è sviluppato costantemente da Oracle, così da restare al passo con le nuove richieste tecnologiche e aziendali. La connettività, velocità, e sicurezza di MySQL lo rendono altamente qualificato per l'accesso ai database su Internet. Alcuni dei principali benefici di MySQL includono:

- **Facilità d'uso:** gli sviluppatori possono installare MySQL nel tempo di qualche minuto e il database è facile da gestire.
- **Affidabilità:** MySQL è uno dei database più maturi e ampiamente utilizzati, ed è stato testato in una grande varietà di scenari per quasi 30

anni, incluse molte delle aziende più grosse al mondo, le quali dipendono da MySQL per eseguire applicazioni critiche, data la sua affidabilità.

- **Scalabilità:** MySQL è in grado di scalare per soddisfare i bisogni delle applicazioni con il maggior numero di accessi. La sua architettura a replicazione nativa permette alle aziende, tra cui Facebook, Netflix e Uber, di scalare le applicazioni per supportare decine di milioni di utenti, se non di più.
- **Prestazioni:** MySQL è un sistema di database ad alte prestazioni e con zero necessità di amministrazione, viene rilasciato in diverse edizioni per soddisfare quasi tutte le richieste.
- **Elevata disponibilità:** MySQL offre un insieme completo di tecnologie di replicazione native e pienamente integrate per permettere una grande disponibilità e al contempo il recupero dei disastri. Per le applicazioni di business critiche e gli accordi di servizio, i clienti possono raggiungere l'obiettivo di zero perdite di dati, con un tempo di recupero istantaneo.
- **Sicurezza:** la sicurezza dei dati riguarda sia la protezione degli stessi dati, che il rispetto delle leggi dell'industria e del governo, incluso il GDPR nell'Unione Europea. MySQL Enterprise è in grado di offrire funzionalità di sicurezza avanzate, come l'autenticazione/autorizzazione, la crittografia trasparente dei dati, l'auditing, il data masking e un firewall per il database.
- **Flessibilità:** il Document Store di MySQL offre agli utenti una massima flessibilità nel sviluppare tradizionali applicazioni database che utilizzino sia SQL che NoSQL, queste ultime senza schemi. Gli sviluppatori possono mischiare tabelle relazionali con documenti JSON nello stesso database e nella stessa applicazione.

2.11.3 Utilizzo nelle applicazioni web

I casi d'uso di MySQL includono la gestione di dati dei clienti e dei prodotti per i siti di e-commerce, aiutando i sistemi di gestione dei contenuti a servire contenuti web, tracciare le transazioni in modo sicuro e i dati finanziari, alimentando i siti di social network salvando i profili degli utenti e le loro interazioni.

L'abilità di MySQL di gestire grandi dataset e query complesse lo rende una tecnologia chiave nelle industrie e i casi d'uso includono:

- E-commerce: molte delle piattaforme di ecommerce più grandi al mondo, come Uber e Booking.com, eseguono i loro sistemi transazionali su MySQL. Costituisce quindi una scelta popolare per gestire i profili utente, le credenziali, il contenuto degli utenti e i dati finanziari, inclusi pagamenti e rilevamento di truffe.
- Piattaforme social: Facebook, X (ovvero l'ex-Twitter) e LinkedIn sono tra le piattaforme social più grandi al mondo e tutti si appoggiano a MySQL per la parte di database.
- Gestione del Contenuto: a differenza dei documenti di database *single-purpose*, MySQL offre l'abilità di utilizzare sia SQL che NoSQL all'interno dello stesso database. Il MySQL Document Store permette operazioni CRUD e utilizza il potere di SQL per effettuare data queries sui documenti JSON per riportarne contenuto e statistiche.
- SaaS: MySQL è il database sottostante a molte applicazioni popolari di *Software-as-a-Service*, come ad esempio Zendesk [42].

2.11.4 Normalizzazione dei dati

La normalizzazione è il processo di organizzazione dei dati all'interno di un database. Include la creazione di tabelle e lo stabilimento di relazioni tra queste in base a regole definite sia per proteggere i dati, sia per rendere il database più flessibile, eliminando la ridondanza e le dipendenze non coerenti.

I dati ridondanti sprecono spazio su disco e danno luogo a problemi di mantenimento. Se abbiamo bisogno di cambiare dei dati che esistono in più di un posto, tutti i dati vanno modificati nella stessa identica maniera in tutte queste locazioni. Una modifica è invece più facile da implementare se i relativi dati esistono solo in un punto e da nessun'altra parte nel database. Una dipendenza incoerente rende i dati difficili da accedere, in quanto il percorso per trovare tali dati potrebbe essere mancante o malfunzionante.

Esistono alcune regole per la normalizzazione di un database e ciascuna di essere è chiamata una *forma normale*. Generalmente si applicano le prime tre regole per ottenere un database in terza forma normale.

Non sempre però si riesce a rispettare tali regole alla lettera: in generale, la normalizzazione richiede la creazione di tabelle aggiuntive e questo talvolta può risultare scomodo. In questo modo, tuttavia, si generano possibili problemi quali dati ridondanti e dipendenze funzionali [43].

2.12 Apache HTTP Server

Apache HTTP Server, detto anche semplicemente Apache, è un web server gratuito e open-source che distribuisce contenuti web attraverso la rete Internet, ed è rapidamente diventato il più popolare client HTTP sul web. Il nome trova le sue origini nel rispetto delle tribù dei nativi americani, per la loro resilienza e durabilità.

Apache è solo uno dei componenti richiesti nello stack di un'applicazione web per la distribuzione del contenuto. Uno degli stack più comuni per le webapp è LAMP, ovvero Linux, Apache, MySQL e PHP, o il suo analogo WAMP che utilizza Windows invece che Linux. Il sistema operativo, Linux o Windows, gestisce le operazioni dell'applicazione. Apache è il web server che processa le richieste e serve risorse e contenuti web tramite protocollo HTTP. MySQL è il database che contiene tutte le informazioni in un formato che permette di eseguire queries facilmente. PHP è il linguaggio di programmazione che funziona con Apache per aiutare a creare contenuto web dinamico.

Anche se le statistiche effettive possono variare, gran parte delle webapp vengono eseguite su una qualche forma di stack LAMP in quanto è facile da costruire e utilizzabile in modo gratuito. Generalmente, la maggior parte delle webapp tende ad avere un'architettura simile, anche se il loro scopo può variare considerevolmente, e possono beneficiare dell'utilizzo di Firewalls, Load Balancers, Web Servers, Content Delivery Networks (CDN) e Database Servers.

I Firewall aiutano a proteggere l'applicazione web sia da minacce esterne che da vulnerabilità interne, in base a come sono configurati. I Load Balancers aiutano a distribuire il traffico tra i server web che possono gestire le richieste HTTP(S), e qui è dove Apache entra in gioco, o le richieste verso application servers, ovvero i server che gestiscono le funzionalità e il carico di lavoro di una webapp. I Database Servers, invece, si occupano dello storage di risorse e backups. In base all'infrastruttura, il database e l'applicazione possono coesistere sullo stesso server, anche se tendenzialmente sarebbe raccomandabile mantenerli separati.

La rete Internet consiste di molte tecnologie differenti e non tutte sono le stesse. Anche se Apache costituisce senza dubbio uno dei server web più popolari che esistono sulla rete, ci sono molti altri competitor e il panorama è in costante cambiamento. Negli anni '90 e primi 2000, Apache dominava il mercato, servendo più del 50% dei siti web attivi su Internet. Un'altra opzione era IIS (Internet Information Services) di Microsoft, ma in confronto era meno popolare. Ad oggi, Apache continua a servire una grande porzione di siti web attivi ma la sua quota di mercato si è ridotta dal 50% a poco sotto il 40% nel 2018 e NGINX, pur essendo un nuovo web server, si trova al secondo posto con circa il 35%, mentre Microsoft IIS si aggira tra l'8 e il 10%. Ogni anno vengono rilasciate nuove webapp con nuovi stack e server, quindi il panorama continua a cambiare.

Apache è considerato un software open source, il che vuol dire che il codice originale è disponibile liberamente per la visione e la collaborazione. Questo lo ha reso molto popolare tra gli sviluppatori, i quali hanno costruito

e configurato dei loro moduli per applicare specifiche funzionalità e migliorare quelle già esistenti. Apache è nato nel 1995 ed è responsabile di aver aiutato ad iniziare la crescita di Internet quando era ancora agli albori.

Uno dei vantaggi di Apache è la sua abilità di gestire grandi quantità di traffico con una configurazione minima. Inoltre, è in grado di scalare con facilità e, grazie alla sua funzionalità modulare, permette di essere configurato in base alle proprie esigenze. Possiamo anche rimuovere moduli indesiderati per rendere Apache più leggero ed efficiente.

Alcuni dei moduli più popolari che possono essere aggiunti includono SSL, il supporto alla programmazione lato server (PHP), e le configurazioni di Load Balancing per gestire grandi quantità di traffico. Apache può essere rilasciato su Linux, macOS e Windows, seppur con percorsi processi di installazione diversi.

Altre funzionalità di Apache Web Server includono la gestione di file statici, il caricamento di moduli dinamici, l'indicizzazione automatica, la configurazione di .htaccess, la compatibilità con IPv6, il supporto a HTTP/2, le connessioni FTP, la compressione e decompressione mediante Gzip, il restringimento della larghezza di banda, il tracciamento della sessione, il load balancing, la riscrittura di URL e la geolocalizzazione basata su indirizzo IP.

Apache comunica sulla rete da client a server utilizzando il protocollo TCP/IP e può essere utilizzato per una grande varietà di protocolli, di cui il più comune è HTTP/S, ovvero *HyperText Transfer Protocol (Secure)*, uno dei principali protocolli sul web.

Il server Apache è configurato tramite file di configurazione, nei quali si utilizzano i moduli per controllarne il comportamento. Di default, Apache si mette in ascolto sugli indirizzi IP configurati nei file di config che si stanno richiedendo. Qui entra in gioco uno dei punti di forza di Apache: mediante la direttiva Listen, è in grado di accettare e ridirezionare traffico specifico per certe porte e domini in base a specifiche combinazioni di richiesta indirizzo-porta. Di default, Listen opera sulla porta 80 (HTTP), ma Apache può essere collegato a diverse porte per diversi domini, permettendo a molti siti

web e domini diversi di essere messi in hosting sullo stesso server. Una volta che un messaggio raggiunge la sua destinazione, invia un messaggio ACK al mittente, indicano che i dati sono arrivati con successo. Se invece avviene un errore nei dati ricevuti, o dei pacchetti sono stati persi durante il transito, allora viene inviato un messaggio NACK che informa il mittente che è necessario ritrasmettere i dati.

Apache HTTP web server è utilizzato da oltre il 67% dei web server in tutto il mondo, dato che si tratta di ambienti facili da personalizzare, ma anche veloci, affidabili e altamente sicuri. Questo ha reso i server Apache una scelta comune anche nelle migliori aziende [44].

2.13 XAMPP

XAMPP (*X per Cross-Platform, Apache, MySQL, PHP e Perl*) è uno dei web server multiplatforma più utilizzati e aiuta gli sviluppatori a creare e testare i loro programmi su un web server locale. Venne sviluppato da Apache Friends e il suo codice, open source, può essere revisionato o modificato dal pubblico. Consiste di un Apache HTTP Server, MariaDB/MySQL e un interprete per i diversi linguaggi di programmazione come PHP e Perl.

XAMPP aiuta un host locale o un server a testare il suo sito e client tramite computer e laptop prima di rilasciarlo sul server principale. Questa piattaforma fornisce un ambiente adatto a testare e verificare il corretto funzionamento dei progetti basati su Apache, Perl, database MySQL e PHP tramite il sistema dello stesso host. Tra tutte queste tecnologie, Perl è un linguaggio di programmazione utilizzato per lo sviluppo web, PHP è un linguaggio di scripting per il backend, e MariaDB è il database più utilizzato sviluppato da MySQL.

2.13.1 Componenti

XAMPP è utilizzato per simboleggiare la classificazione di soluzioni per tecnologie differenti. Fornisce una base per il testing di progetti basati su

diverse tecnologie attraverso un server personale. XAMPP è un insieme di software che contiene un web server chiamato Apache, un sistema di gestione di database chiamato MariaDB e linguaggi di scripting e programmazione come PHP e Perl. Può funzionare su Windows, Linux e macOS.

- Cross-Platform: diversi sistemi locali hanno diverse configurazioni dovute ai sistemi operativi che vi sono installati. I componenti multipiattaforma sono inclusi per aumentare l'utilità e il pubblico a cui si rivolge questa distribuzione, supportando varie piattaforme come Windows, Linux e macOS.
- Apache: si tratta di un web server HTTP multipiattaforma, utilizzato in tutto il mondo per la distribuzione di contenuto web. L'applicazione server è stata resa gratuita per l'installazione ed è utilizzata dalla community di sviluppatori sotto l'egida di Apache Software Foundation. Il server remoto di Apache consegna i file richiesti, le immagini e altri documenti all'utente.
- MariaDB: in origine, il DBMS MySQL era parte di XAMPP, ma ad oggi è stato sostituito da MariaDB. Questo è uno dei DBMS più utilizzati, sviluppato da MySQL, e offre servizi online di data storage, manipolazione dati, recupero dati, riordinamento e cancellazione.
- PHP: è il linguaggio di scripting per backend principalmente utilizzato per lo sviluppo web. PHP permette agli utenti di creare siti web e applicazioni dinamiche. Può essere installato su ogni piattaforma e supporta una grande varietà di DBMS (*Database Management Systems*). Inoltre, è stato implementato utilizzando il linguaggio C. Il nome PHP è stato derivato da *Personal Home Page tools*, il che spiega la sua semplicità e funzionalità.
- Perl: è una combinazione di due linguaggi dinamici ad alto livello, ovvero Perl 5 e Perl 6. Perl può essere applicato per trovare soluzioni a

problemi basati sull'amministrazione di sistema, sviluppo web e networking. Perl permette ai suoi utenti di programmare applicazioni web dinamiche ed è molto flessibile e robusto.

- phpMyAdmin: questo strumento è utilizzato per gestire MariaDB e si occupa dell'amministrazione di DBMS.
- OpenSSL: è l'implementazione open-source del protocollo *Secure Socket Layer / Transport Layer Protocol* (SSL / TLS).
- XAMPP Control Panel: questo pannello aiuta a operare e regolare altri componenti di XAMPP.
- Webalizer: questa soluzione software di Web Analytics è utilizzata per i log utenti e fornisce dettagli sull'utilizzo.
- Mercury: è un sistema di trasporto di email, ovvero un mail server, che aiuta a gestire le email attraverso il web.
- Tomcat: è un servlet basato su Java che fornisce funzionalità di Java.
- Filezilla: è un server avente protocollo FTP (*File Transfer Protocol*), che supporta e facilita le operazioni di trasferimento eseguite sui file.

2.14 Git

Git è uno strumento di controllo delle versioni del codice (VCS, ovvero *Version Control System*) che è diventato pressoché un must negli ecosistemi di sviluppo software. L'abilità di Git di tracciare meticolosamente i cambiamenti a un progetto lo rende uno strumento essenziale per gli sviluppatori che mirano a gestire i loro progetti in modo efficiente.

Il controllo delle versioni (*version control*) ci permette di tracciare i cambiamenti al codice di un software. Quindi, la versione distribuita di un software consiste in un insieme di versioni specifiche di ciascuno dei suoi

componenti e file di codice, in quanto ciascuno di essi potrebbe essere stato modificato un numero estremamente variabile di volte.

Tenere traccia dei cambiamenti al codice ci permette di rendere più facile l'identificazione dell'origine di un problema, e allo stesso tempo riduce il rischio di conflitti e sovrascrittura di file. Quindi, Git facilita e semplifica il versioning del software precisamente per questo motivo.

Git offre alcune funzionalità chiave che rendono più facile ottimizzare la gestione del codice e la collaborazione tra teams.

Visualizzazione della storia del progetto

La storia dei *commit* è un pilastro chiave per tracciare i progressi del progetto su Git ed è quindi il motivo per cui Git offre agli sviluppatori uno storico dettagliato di tutti i cambiamenti apportati al codice.

Per ciascun nuovo commit vengono tracciati gli specifici cambiamenti apportati ai file del progetto, insieme a un breve messaggio di spiegazione scritto dallo sviluppatore che ha effettuato quel cambiamento. Questi elementi aiutano a migliorare la capacità di comunicazione del team, permettendo una comprensione più rapida delle *insertions* e *deletions* che avvengono nel codice.

In aggiunta al monitoraggio dello sviluppo, questo storico ci permette di tornare indietro se necessario, cancellando parte dei cambiamenti oppure effettuando la *fetch* di solo una parte dei cambiamenti da un *branch* all'altro. Questa funzione svolge un ruolo essenziale nel mantenimento della trasparenza, coerenza e qualità del codice di un progetto su Git, insieme al miglioramento della collaborazione all'interno del team di sviluppo e dell'efficienza operativa per risolvere i problemi.

Maggiore autonomia per i teams

Un'altra funzionalità essenziale di Git è lo sviluppo distribuito. Grazie alla sua struttura decentralizzata, Git permette ai team di sviluppatori di lavorare simultaneamente allo stesso progetto, fornendo a ogni membro una

propria copia del progetto dove ciascun cambiamento apportato può essere tracciato nel versioning. Questo permette loro di lavorare in autonomia su funzionalità specifiche e di ridurre i rischi di conflitti o sovrascrittura. Questo approccio offre una grande flessibilità per gli sviluppatori che possono quindi esplorare idee diverse e sperimentare nuove funzionalità senza interferire col lavoro svolto dai loro colleghi.

Lo sviluppo distribuito migliora anche la resilienza ai guasti del server: in caso di server failure, ciascuna persona possiede una copia del codice su cui possono continuare a lavorare offline. I cambiamenti possono quindi essere sincronizzati una volta che il server ritorna nuovamente disponibile, riducendo il rischio di distruzione del lavoro per i team di sviluppo e i limiti di aggiornamento per i team operazionali.

Ottimizzazione dei workflow di sviluppo

Git è in grado di gestire *branches* e di effettuare il *merging*, permettendo ai team di lavorare in parallelo in un modo collaborativo e organizzato. Ogni nuova aggiunta al codice o bugfix può essere sviluppata e testata indipendentemente, per garantire l'affidabilità. Gli sviluppatori quindi possono semplicemente effettuare il *merge* di questi cambiamenti nel main branch del progetto.

Adottando questo approccio, i team possono tracciare l'evoluzione del codice, collaborare facilmente e in modo efficiente, riducendo i conflitti tra versioni differenti e assicurando un'integrazione continua delle funzionalità sviluppate. I teams possono così sviluppare progetti in modo continuativo e agile, mentre rilasciano regolarmente nuove versioni di codice. Questa pratica facilita molto la gestione dei cambiamenti e allo stesso tempo riduce il rischio di errori.

2.14.1 Vantaggi di Git

Git offre numerosi benefici:

- Gestione del versioning decentralizzata: con Git, ogni sviluppatore possiede una copia completa della storia del progetto, permettendo loro di lavorare indipendentemente.
- Sicurezza: a differenza di altri VCS, Git assicura l'integrità di tutti gli elementi all'interno del repository mediante un algoritmo crittografico di hash detto *Secure Hash Algorithm* (SHA-1 e SHA-256). Questo algoritmo protegge il codice e la storia del progetto da ogni modifica benevola o malevola. Inoltre, ogni commit (creazione di una nuova versione) può essere automaticamente firmato (GPG) per assicurare la tracciabilità dei cambiamenti. Questo rende Git uno strumento particolarmente sicuro e affidabile, garantendo l'integrità e l'autenticità del codice e della sua storia.
- Veloce ed efficiente: Git massimizza l'efficienza durante lo sviluppo e la sua velocità permette agli sviluppatori di svolgere operazioni complesse, come i commit, branching e merging, in un tempo minimo, anche su grandi codebase. Assicura inoltre un impatto minimo sull'hard disk e anche durante gli scambi su rete. Questa efficienza si traduce in tempi di risposta rapidi durante i backup, le consultazioni e i cambiamenti della storia del progetto.
- Maggiore flessibilità di lavoro: Git supporta una grande varietà di workflow di sviluppo, dai modelli centralizzati fino agli approcci lineari. Questa abilità di gestire diversi workflow fornisce ai team numerose opzioni per le possibilità con cui possono lavorare.
- Facilità di integrazione: Git eccelle nella sua abilità di integrarsi con una grande varietà di strumenti e piattaforme di sviluppo. Questa larga compatibilità permette ai team di gestire i progetti più efficientemente, sfruttando i migliori tool e pratiche di DevSecOps.
- Progetto open-source molto popolare: Git viene supportato da una community dinamica e dedicata, che assicura il suo miglioramento co-

stante. Questa partecipazione attiva da individui e aziende garantisce la regolare aggiunta di nuove funzionalità e miglioramenti attraverso aggiornamenti continui.

2.14.2 Principali comandi di Git

Git offre una grande varietà di comandi per rendere più facile il lavoro di squadra, quelli più comuni sono:

- `git init` inizializza un nuovo repository Git.
- `git clone [url]` clona un repository esistente.
- `git add[file]` aggiunge un file all'indice.
- `git commit` valida i cambiamenti apportati.
- `git commit -m "message"` valida i cambiamenti con un messaggio.
- `git status` visualizza lo status dei file nella *working directory*.
- `git push` invia i cambiamenti al repository remoto.
- `git pull` effettua la *fetch* dei cambiamenti dal repository remoto e quindi effettua la *merge* con quelli del repository locale [45].

2.14.3 Piattaforme di hosting per repositories

Un repository Git può essere messo in hosting in diverse piattaforme, tra cui le principali sono GitHub, GitLab e BitBucket, le quali ospitano repository Git rendendoli accessibili pubblicamente e offrono anche molte delle stesse funzionalità utilizzabili dal terminale con una GUI più user-friendly.

Visualizzazione dati sull'utilizzo del progetto BolognaWiFi

In questo capitolo andremo a trattare l'applicazione sviluppata, partendo da alcune considerazioni sui dati ufficiali forniti pubblicamente dal Comune di Bologna, per arrivare ad analizzare la struttura della stessa webapp.

Approfondiremo quindi i principali obiettivi, i problemi affrontati, le soluzioni implementate e l'ambiente di lavoro impiegato, per mostrare infine qualche schermata dell'applicazione così realizzata.

3.1 Il caso studio: dati del BolognaWiFi

Il Comune di Bologna offre una serie di Open Data relativi al BolognaWiFi, riguardanti affollamento, affluenza e spostamenti, che vengono pubblicati ogni giorno mediante la rete Internet, in formato JSON. Il loro contenuto si riferisce a tutte le entrate giornaliere, riguardo alle varie zone e spostamenti, aggiornate a tre giorni prima della data odierna.

Tutti questi dati sono stati aggregati in maniera oraria e anonima, quindi è impossibile risalire ai singoli dispositivi, rispettando così la privacy degli utenti. Tuttavia, la struttura dei dati non è omogenea in tutti e tre i dataset, quindi andremo a esplorarla nel dettaglio.

3.1.1 Spostamenti

Per gli spostamenti viene registrata una nuova entrata per ogni spostamento che viene effettuato in una data ora del giorno. Non è detto che ad ogni ora del giorno corrisponda almeno uno spostamento da una zona all'altra, in quanto non è detto che avvengano sempre, si pensi ad esempio durante le ore notturne.



```
{
  "total_count": 1134133,
  "results": [
    {
      "data_evento": "2021-05-18",
      "hour": 16,
      "giorno": "2-Martedì",
      "percentile_50": 6,
      "tot_pass": 13,
      "area_to": "mambo",
      "area_from_label": "Parco del Cavaticcio",
      "area_from": "parco_cavaticcio",
      "area_to_label": "Mambo"
    },
    ...
  ]
}
```

Figura 3.1: Dati relativi agli spostamenti, in formato JSON, riferiti ai movimenti da Parco del Cavaticcio a Mambo, alle ore 16 del 18 maggio 2021.

Tali spostamenti sono direzionati, quindi uno spostamento dalla Biblioteca Salaborsa a Palazzo D'Accursio verrà registrato separatamente da quello effettuato, inversamente, da Palazzo D'Accursio a Biblioteca Salaborsa. Anche in questo caso, entrambe le direzioni degli spostamenti sono indipendenti l'una dall'altra, e non è detto che esistano entrambe in una certa ora. Per esempio, potremmo avere una direzione in cui si spostano 100 persone (o meglio, dispositivi), mentre nell'altra se ne spostano 0 o comunque una quantità trascurabile ai fini del rilevamento dati di BolognaWiFi, quindi in entrambi questi ultimi due casi non verrebbe creata nessuna nuova entrata sul data-

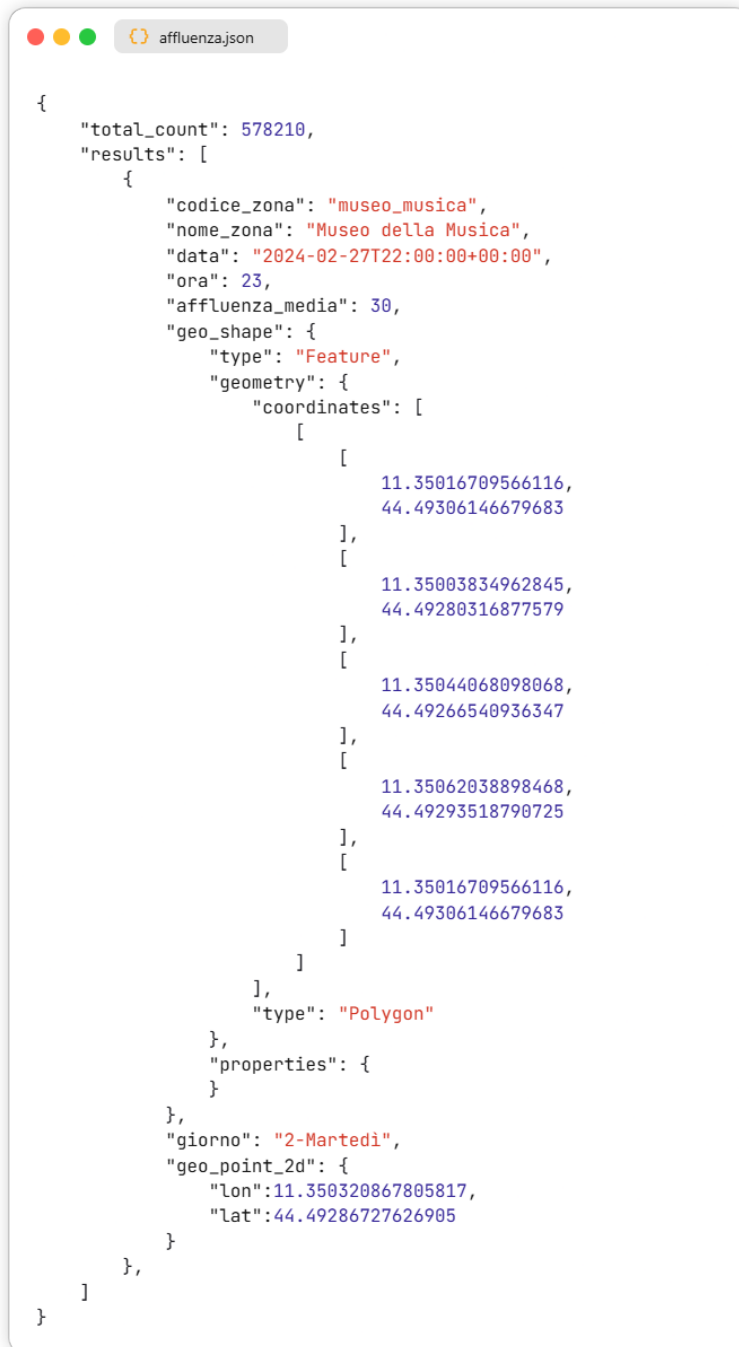
base di Open Data. Questo aiuta anche a garantire la privacy degli utenti, in quanto se venissero pubblicati i dati di un singolo dispositivo in una certa direzione il diritto alla privacy potrebbe venire violato.

3.1.2 Affollamento e Affluenza

Dato che affollamento e affluenza contengono dati aventi una struttura molto simile tra loro, possiamo trattarli insieme. Per entrambi viene registrata una nuova entrata per ogni ora del giorno, in ciascuna zona di Bologna coperta dal servizio BolognaWiFi.


```
{
  "total_count": 553688,
  "results": [
    {
      "codice_zona": "piazza_malpighi",
      "nome_zona": "Piazza Malpighi",
      "data": "2024-02-12T19:00:00+00:00",
      "giorno": "1-Lunedì",
      "ora": 20,
      "affollamento_medio": 230,
      "geo_shape": {
        "type": "Feature",
        "geometry": {
          "coordinates": [
            [
              [
                11.33636176586151,
                44.49439885825328
              ],
              [
                11.33603990077972,
                44.49375599580188
              ],
              [
                11.33632957935333,
                44.49369094384946
              ],
              [
                11.33667826652527,
                44.49434528665299
              ],
              [
                11.33636176586151,
                44.49439885825328
              ]
            ]
          ]
        },
        "type": "Polygon"
      },
      "properties": {
      },
      "geo_point_2d": {
        "lon": 11.336354718343932,
        "lat": 44.494051038094796
      }
    },
    ...
  ]
}
```

Figura 3.2: Dati riguardanti l'affollamento, in formato JSON, riferiti a Piazza Malpighi alle ore 20 del 12 febbraio 2024. Notare l'attributo *ora* che viene incluso nel timestamp di *data* in fuso orario UTC.



```
{
  "total_count": 578210,
  "results": [
    {
      "codice_zona": "museo_musica",
      "nome_zona": "Museo della Musica",
      "data": "2024-02-27T22:00:00+00:00",
      "ora": 23,
      "affluenza_media": 30,
      "geo_shape": {
        "type": "Feature",
        "geometry": {
          "coordinates": [
            [
              [
                11.35016709566116,
                44.49306146679683
              ],
              [
                11.35003834962845,
                44.49280316877579
              ],
              [
                11.35044068098068,
                44.49266540936347
              ],
              [
                11.35062038898468,
                44.49293518790725
              ],
              [
                11.35016709566116,
                44.49306146679683
              ]
            ]
          ],
          "type": "Polygon"
        },
        "properties": {}
      },
      "giorno": "2-Martedì",
      "geo_point_2d": {
        "lon": 11.350320867805817,
        "lat": 44.49286727626905
      }
    }
  ]
}
```

Figura 3.3: Dati riguardanti l'affluenza, in formato JSON, riferiti al Museo della Musica, alle ore 23 del 27 febbraio 2024. Notare l'attributo *ora* che viene incluso nel timestamp di *data* in fuso orario UTC.

3.2 Struttura del database

La struttura dei dati ottenuti dall'interrogazione delle API, così come possiamo vedere in Figura 3.1, Figura 3.2 e Figura 3.3, è abbastanza semplice di per sé. Tuttavia, tale semplicità nasconde dei problemi di gestione delle ridondanze quando vengono salvati i dati su database, ciò avviene per svariati motivi che ora approfondiremo.

3.2.1 Aree

Innanzitutto, le informazioni relative alle aree e coordinate sono salvate in maniera differente nei tre dataset. Negli spostamenti si collegano solo i nomi e gli identificatori di ciascuna area attinente al vettore movimento, mentre sia in affollamento che in affluenza vengono specificate anche le relative coordinate del poligono, insieme a quelle del suo centroide.

Questo comporta fin da subito la necessità di creare un'entità per le aree e una separata per ciascuna coordinata, in quanto ogni poligono ha un numero variabile di punti. In questo modo non solo rimuoviamo la ridondanza tra affollamento e affluenza, ma permettiamo anche di collegare due poligoni a ciascuno spostamento, uno di partenza e uno di arrivo, cosa che non sarebbe possibile se non avessimo aggregato i tre dataset.

Notiamo subito che tale accortezza è possibile solamente perché tutti e tre i dataset afferiscono alle stesse aree, ovvero i poligoni, delimitati dalla copertura del segnale di BolognaWiFi, quindi il loro dominio delle aree è completamente sovrapponibile.

3.2.2 Spostamenti

In questo modo, abbiamo assegnato un poligono di partenza e uno di arrivo a ciascuno spostamento. Rimane tuttavia il fatto che esista un numero n di spostamenti giornalieri, con $n \leq 24$, quindi al momento avremmo un

numero esageratamente alto di entrate su database, che diventerebbe un problema in termini di richieste al server per visualizzare i dati giornalieri.

Questo problema può essere efficacemente risolto aggregando tutti i dati giornalieri: per fare ciò, fissata un'area di partenza e una di arrivo per una precisa data, creiamo 24 colonne per l'attributo *percentile_50* e altrettante 24 per *tot_pass*. In questo modo ottimizziamo il recupero dei dati effettuato dal client mediante richieste server, riducendo il numero di chiamate fino a 24 volte: basta semplicemente scaricare tutti i dati di un giorno, e utilizzarli fino a quando l'utente non chiama i dati di un altro giorno.

Riguardo al significato degli attributi, *tot_pass* è il numero di passeggeri che transitano da una zona A a una zona B nell'arco di un'ora in un determinato giorno. Invece, *percentile_50* è il mediano di tutti i *tot_pass* calcolati per tutti i giorni a una determinata ora, fino al giorno in cui vengono salvati i dati sul database di Open Data.

In questo caso, l'aggregazione dei dati è semplice perché nel dataset, nessun giorno ha un numero di entrate superiore a 24, cosa non scontata considerando che il cambio dell'ora genera un giorno con 23 ore in primavera e uno con 25 in autunno. Tale affermazione è affidabile in quanto la registrazione dei dati sugli spostamenti è cominciata l'1 aprile 2021.

3.2.3 Affollamento e Affluenza

Ancora una volta, questi due dataset si rivelano simili: avendo un insieme di attributi molto simili e lo stesso set di coordinate dei poligoni, basta scegliere uno qualunque dei due dataset per popolare le tabelle di area e coordinate che abbiamo visto prima.

In questo modo, entrambe le sezioni di *geo_shape* e *geo_point_2d* vengono assorbite in area e coordinate, alleggerendo la struttura delle tabelle di affollamento e affluenza che andremo a realizzare. Possiamo notare che tutti i dati relativi ad affollamento e affluenza hanno un *"type"*: *"Feature"* e *"type"*: *"Polygon"*, che possono essere tranquillamente ignorati essendo uguali in tutto il dataset.

Anche qui, per ottimizzare le richieste al server nello stesso modo che abbiamo visto per gli spostamenti, aggregiamo le 24 ore dello stesso giorno all'interno della stessa entrata, ma non solo: data la loro somiglianza, affollamento e affluenza possono essere aggregati all'interno della stessa tabella. Questo è possibile perché entrambe le loro entrate riguardano tutte le aree di Bologna ad ogni ora del giorno, quindi il numero totale è sempre uguale.

Tuttavia, in questo caso in entrambi i dataset vi sono giorni con 23 o 25 entrate giornaliere, a causa del cambio dell'ora. Tale problema si risolve facilmente, considerando che tutti gli eccessi hanno valori nulli: in caso di conflitto, basta tenere il valore massimo, in quanto uno dei due è per forza zero, mentre in caso di deficit basta assegnare 0 come valore al dato mancante.

Inoltre, sorge un ulteriore problema iniziale dovuto al fatto che i dati di affluenza sono disponibili a partire dal 1 gennaio 2024, mentre quelli riguardanti l'affollamento cominciano dal 15 gennaio 2024. Ancora una volta si risolve semplicemente assegnando uno 0 ai valori mancanti, ovvero a tutti quelli nelle due settimane in cui non vi sono dati riguardanti l'affollamento.

3.2.4 Schema finale

Dopo tutte le precedenti considerazioni, riportiamo in Figura 3.4 lo schema finale del database. Per evitare di ottenere un database enorme in termini di spazio su disco, abbiamo prestato particolare attenzione ad utilizzare la quantità di memoria minore possibile per ogni attributo delle tabelle.

In particolare, considerando che gli spostamenti (sia totali che mediani), l'affollamento e l'affluenza orari raggiungono valori sempre interi che non superano mai la quantità di 10 000 e non sono mai inferiori a zero. Di conseguenza, possiamo utilizzare un tipo di dato *unsigned short* senza avere timori relativi ad un possibile overflow, anzi possiamo mantenere pure una certa quantità di margine, occupando solo 2 bytes per numero, dimezzando lo spazio richiesto in confronto ai 4 bytes occupati da un normale *unsigned int*.

Inoltre, una simile considerazione può essere effettuata per i numeri decimali relativi alle coordinate dei punti dei poligoni, in quanto vi sono sempre due cifre intere e 14 o 15 cifre decimali. Riguardo alle due cifre intere, possiamo utilizzarne sempre due solamente considerando la posizione geografica di Bologna, in quanto se la latitudine non rappresenta un problema, la longitudine talvolta può assumere valori a tre cifre intere in base alla posizione sulla Terra. In questo modo, su MySQL gli viene assegnato il tipo di dato *decimal(17,15)*, con 15 cifre decimali su 17 cifre totali, escludendo la virgola.

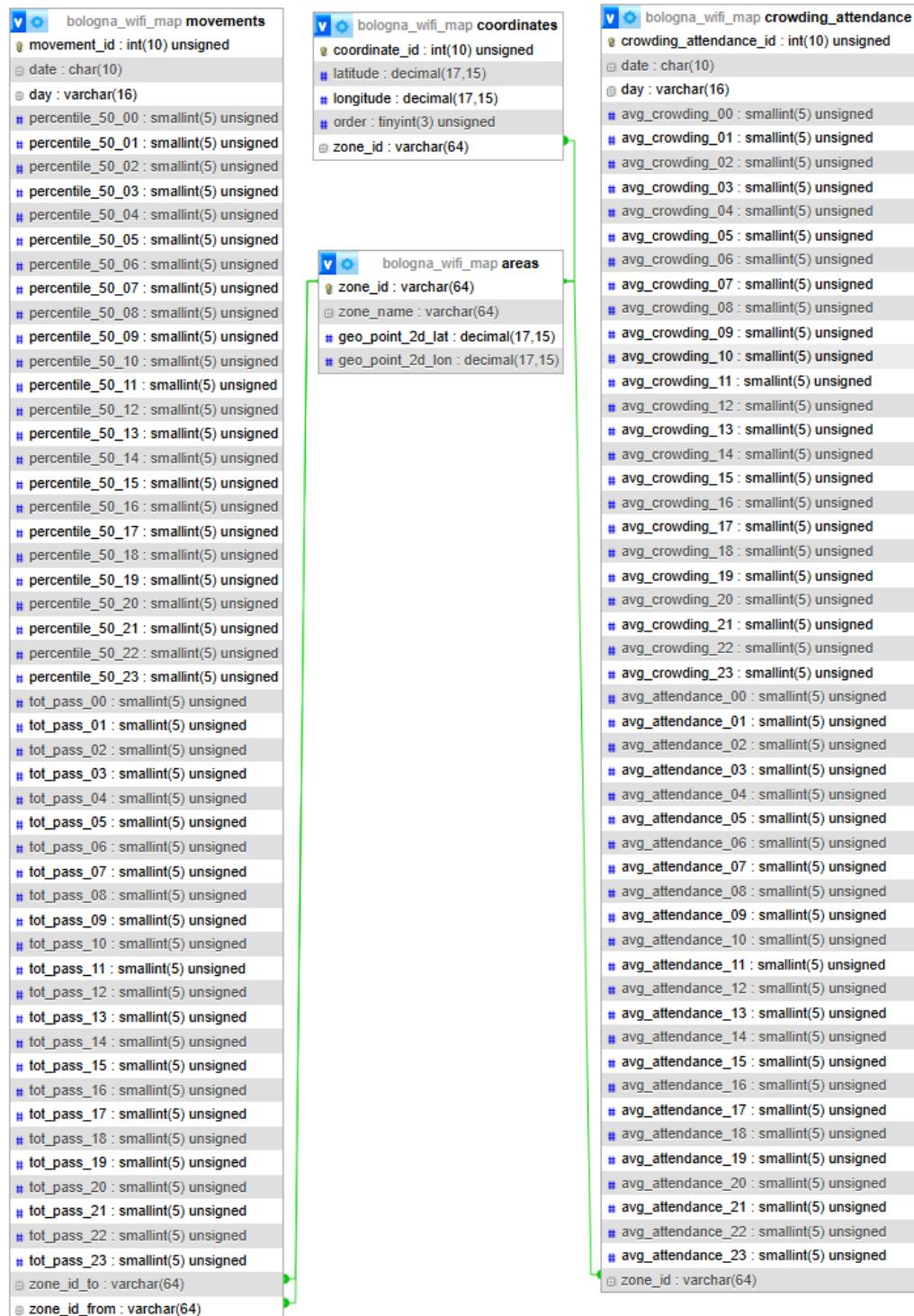


Figura 3.4: Schema del database di BolognaWiFiMap. Notare la lista di attributi definiti per ciascuna ora.

3.3 Connessione tra frontend e backend

Dopo aver installato Vue.js e creato il progetto utilizzando Vite, ovvero un plugin di Vue per la nuova sintassi SFC (Single-File Components), abbiamo impostato l'indirizzo e la porta del server nel file di configurazione `vite.config.js`, settandolo a 0.0.0.0 per permettere al frontend di comunicare col backend indipendentemente dall'indirizzo del server, che in questo caso si riferisce a quello del server Apache. Inoltre, abbiamo specificato l'alias `@` per definire la directory `./src/frontend`, semplificando il percorso dei file da importare.



```
// https://vite.dev/config/
export default defineConfig({
  server: {
    host: "0.0.0.0",
    port: 5173,
  },
  plugins: [
    vue(),
    vueDevTools(),
  ],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src/frontend', import.meta.url))
    },
  },
})
```

Figura 3.5: Configurazione delle impostazioni Vite relative al progetto Vue, dove si definiscono in particolare l'indirizzo e la porta per accedere al server.


3.4 Single-Page App

Questa applicazione è stata realizzata seguendo il modello della *Single-Page App*, ovvero una sola pagina web che carica selettivamente i suoi componenti o aggiorna i suoi dati senza necessità di ricaricare la pagina. Questo è permesso sia dallo stesso framework Vue.js, sia dall'implementazione di richieste AJAX che il client effettua al server.

3.5 Implementazione richieste AJAX

Le richieste AJAX sono effettuate in modo asincrono dal client al server per ottenere dei dati che verranno utilizzati per aggiornare la pagina, evitando di dover effettuare un ricaricamento della stessa. Per la sua implementazione abbiamo utilizzato la funzione `fetch()` built-in di JavaScript, creando tre funzioni per aggiornare rispettivamente le aree, con `fetchAreas()`, gli spostamenti, con `fetchMovements()`, e infine l'affollamento insieme all'affluenza, con `fetchCrowdingAttendance()`.

Abbiamo definito tutte queste funzioni nel file `api.js`, essendo molto semplici riportiamo in Figura 3.6 solo quella relativa agli spostamenti. In particolare, queste custom API utilizzano l'indirizzo corrente comprensivo di porta, per poi appendere a tale stringa l'url specifico dell'API a cui si vuole effettuare la richiesta. Si noti come questo setup presuppone un deployment di frontend e backend sullo stesso server, o quantomeno di un ridirezionamento tramite proxy di tali richieste effettuate a frontend verso l'url appropriato del backend, se dovessero essere in deployment su server diversi.



```
const baseUrl = `${window.location.protocol}//${window.location.hostname}`;

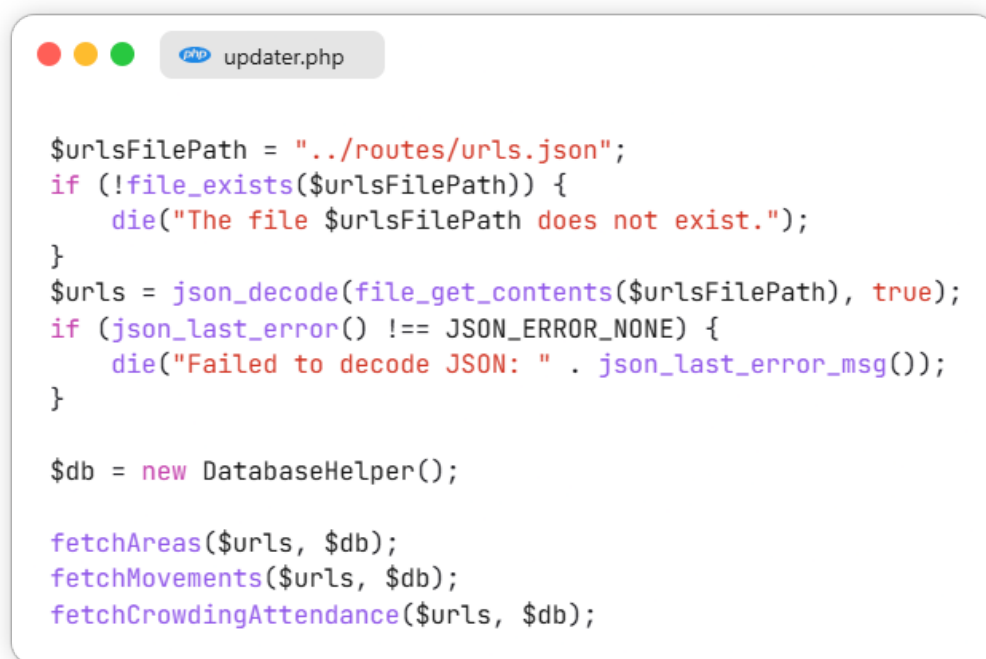
/**
 * Fetch all movements from server for a specific day
 * @param {string} date
 * @param {Ref<Movements[]>} movements
 * @param {Ref<Medians[]>} medians
 * @returns {Promise<void>}
 */
export const fetchMovements = async (date, movements, medians) => {
  try {
    const url = `${baseUrl}/backend/api/movements.php`;
    const response = await fetch(url, {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({date}),
    });
    const data = await response.json();
    if (data.success) {
      /** @type {MovementsMedians} */
      let results = data.results;
      addObjectToKey(movements, date, results.tot_pass);
      addObjectToKey(medians, date, results.percentile_50);
    } else {
      console.error("Failed to fetch movements: ", data.message);
    }
  } catch (error) {
    console.error("Failed to fetch movements: an error occurred.\n");
  }
}
```

Figura 3.6: Implementazione di una funzione che effettua una richiesta AJAX per ottenere la lista dei movimenti di una specifica data.

3.6 Popolamento del database

Affinché le richieste AJAX possano andare a buon fine, bisogna aver popolato il database. Questo compito è svolto mediante l'esecuzione del file `updater.php`, il quale controlla l'ultima data dei dati raccolti sul database e provvede ad aggiornarli, scaricando tutti i nuovi dati.

In fase di development o test, questo script viene lanciato manualmente, ma in un futuro deployment basterebbe automatizzare la sua esecuzione utilizzando Task Scheduler su Windows o un Cron Job su Linux. Tale script, riportato in Figura 3.7 richiama 3 funzioni differenti per svolgere il suo compito, che qui non mostreremo per via della loro lunghezza.



```
$urlsFilePath = "../routes/urls.json";
if (!file_exists($urlsFilePath)) {
    die("The file $urlsFilePath does not exist.");
}
$urls = json_decode(file_get_contents($urlsFilePath), true);
if (json_last_error() !== JSON_ERROR_NONE) {
    die("Failed to decode JSON: " . json_last_error_msg());
}

$db = new DatabaseHelper();

fetchAreas($urls, $db);
fetchMovements($urls, $db);
fetchCrowdingAttendance($urls, $db);
```

Figura 3.7: Script che viene eseguito per effettuare il setup o per aggiornare i dati sul database, utilizzando gli url definiti in *urls.json* per effettuare le chiamate alle API del sito degli Open Data di BolognaWiFi [14, 15, 16].

Innanzitutto, viene aggiornata la lista delle aree, nel caso se ne trovino di nuove. Quindi si scaricano, in ordine, prima i movimenti, poi l'affollamento insieme all'affluenza. All'interno di entrambe le funzioni, vengono scaricati i dati giorno per giorno, fino ad arrivare agli ultimi disponibili. Tuttavia, date le limitazioni delle API offerte da BolognaWiFi, vengono scaricati al massimo 100 elementi alla volta. Questo chunk così ottenuto viene messo

in coda in una lista, per poi effettuare una nuova chiamata alla stessa API riguardante lo stesso giorno, ma con un offset aumentato di 100. Le stesse API pongono un limite totale di 10 100 alla somma di offset e numero di elementi, ma considerando che in un singolo giorno non si arriva nemmeno a 2 000 elementi totali in tutti e tre i dataset, tale limite effettivo di 10 000 all'offset non è poi così restrittivo.

3.7 Informazioni su una zona o spostamento

3.8 Calcolo del colore di una zona

3.9 Raggruppamento dei dati giornalieri

3.10 Questionario SUS sull'esperienza utente

Bibliografia

- [1] Open data handbook - introduzione, 2025. URL <https://opendatahandbook.org/guide/en/introduction/>.
- [2] Google trends, 2025. URL <https://trends.google.com/trends/explore?date=all&q=%2Fm%2F0119t0j6&hl=en>.
- [3] Open data handbook - perché scegliere gli open data, 2025. URL <https://opendatahandbook.org/guide/en/why-open-data/>.
- [4] The open definition, 2025. URL <https://opendefinition.org/>.
- [5] The full open definition, 2025. URL <https://opendefinition.org/od/2.1/en/>.
- [6] Yahoo website scraping tool, 2025. URL <http://developer.yahoo.com/yql/>.
- [7] Bologna smart city, 2025. URL <https://smartcityweb.net/smartcities/bologna>.
- [8] Bologna è sempre più una smart city, 2022. URL <https://www.bolognatoday.it/speciale/bologna-smart-city.html>.
- [9] Rachele Spinelli. Iperbole: una città connessa, 2020. URL <https://compassunibo.wordpress.com/2020/02/13/iperbole-una-citta-connessa/>.

-
- [10] Open data bolognawifi - elenco hotspot, 2025. URL <https://opendata.comune.bologna.it/explore/dataset/bolognawifi-elenco-hot-spot/information/>.
- [11] Smart city e wifi, la piazza più connessa d'europa, 2014. URL <https://www.alfacod.it/blog-smart-city-wifi-bologna-piazza-piu-connessa-europa>.
- [12] Alessandro Biancardi. Bologna si butta su wi-fi e bluetooth, 2005. URL <https://www.punto-informatico.it/bologna-si-butta-su-wi-fi-e-bluetooth/>.
- [13] Informativa relativa al servizio emiliaromagnawifi resa ai sensi e per gli effetti del regolamento eu n. 679/2016 (gdpr), 2023. URL <https://digitale.regione.emilia-romagna.it/emiliaromagnawifi/wifiprivacy>.
- [14] Open data bolognawifi - matrice spostamenti, 2025. URL <https://opendata.comune.bologna.it/explore/dataset/bolognawifi-matrice-spostamenti/information/?disjunctive=hour>.
- [15] Open data bolognawifi - affluenza, 2025. URL <https://opendata.comune.bologna.it/explore/dataset/iperbole-wifi-affluenza/information/>.
- [16] Open data bolognawifi - affollamento, 2025. URL <https://opendata.comune.bologna.it/explore/dataset/iperbole-wifi-affollamento/information/?disjunctive=ora>.
- [17] Open data bolognawifi - connessioni giornaliere, 2025. URL <https://opendata.comune.bologna.it/explore/dataset/bolognawifi-connessioni-giornaliere/information/>.

- [18] Open data bolognawifi - elenco aree segnale, 2025. URL <https://opendata.comune.bologna.it/explore/dataset/bolognawifi-elenco-aree-segnale/information/>.
- [19] Data visualization: consentire alle città di comunicare facilmente con i cittadini, 2016. URL <https://www.3dgis.it/it/data-visualization-consentire-alle-citta-comunicare-facilmente-cittadini/>.
- [20] Che cos'è la visualizzazione dei dati?, 2025. URL <https://www.oracle.com/it/business-analytics/what-is-data-visualization/>.
- [21] Che cosa caratterizza una buona visualizzazione dei dati?, Settembre 2021. URL <https://www.digitalpills.it/buona-visualizzazione-dei-dati#:~:text=La%20visualizzazione%20deve%20essere%20efficace,di%20cui%20si%20sta%20presentando.>
- [22] Open data handbook - formati file, 2025. URL <https://opendatahandbook.org/guide/en/appendices/file-formats/>.
- [23] Open data: una porta aperta verso nuove opportunità, 2024. URL <https://comunefacile.eu/open-data-una-porta-aperta-verso-nuove-opportunita/>.
- [24] Data integration, cos'è e come fare integrazione dei dati, 2024. URL https://blog.osservatori.net/it_it/data-integration-cosa-significa-come-farla.
- [25] Open data handbook - diritti legali, 2025. URL <https://opendatahandbook.org/guide/en/appendices/what-legal-ip-rights-are-there-in-databases/>.
- [26] Html: cos'è, come funziona e a cosa serve, 2025. URL <https://boolean.careers/blog/html-cose-come-funziona-e-a-cosa-serve#:~:text=HTML%20%C3%A8%20un%20acronimo%20che,%2D%20ovvero%2C%20i%20siti%20web.>

-
- [27] Guida css di base, 2013. URL <https://www.html.it/pag/14209/introduzione1/>.
 - [28] What is scss? a beginner's guide for developers, 2024. URL <https://www.upwork.com/resources/what-is-scss>.
 - [29] What's the difference between css, sass, and scss?, 2021. URL <https://dev.to/mathlete/what-s-the-difference-between-css-sass-and-scss-g2b>.
 - [30] What is javascript?, 2024. URL https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/What_is_JavaScript.
 - [31] Ajax, 2024. URL <https://developer.mozilla.org/en-US/docs/Glossary/AJAX>.
 - [32] Which libraries and frameworks available in javascript?, 2024. URL <https://www.geeksforgeeks.org/which-libraries-and-frameworks-available-in-javascript/>.
 - [33] Introduction to node.js, 2025. URL <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>.
 - [34] Leaflet overview, 2024. URL <https://leafletjs.com/>.
 - [35] Vue.js doc, 2025. URL <https://vuejs.org/guide/introduction.html>.
 - [36] An introduction to the npm package manager, 2025. URL <https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager>.
 - [37] What is php and what can it do?, 2025. URL <https://www.php.net/manual/en/introduction.php>.

-
- [38] What is an api (application programming interface)?, 2025. URL <https://aws.amazon.com/what-is/api/>.
- [39] Danny Dig and Ralph Johnson. How do apis evolve? a story of refactoring. *Journal of Software Maintenance and Evolution: Research and Practice*, 2006. URL https://dig.cs.illinois.edu/papers/JSME_API_Evolution.pdf.
- [40] Working with json, 2024. URL https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/JSON.
- [41] What is sql (structured query language)?, 2025. URL <https://aws.amazon.com/what-is/sql/>.
- [42] Mysql: Understanding what it is and how it's used, 2024. URL <https://www.oracle.com/mysql/what-is-mysql/>.
- [43] Description of the database normalization basics, 2024. URL <https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description>.
- [44] What is apache? in-depth overview of apache web server, 2019. URL <https://www.sumologic.com/blog/apache-web-server-introduction/>.
- [45] What is git? the ultimate guide to git's role and functionality, 2024. URL <https://about.gitlab.com/blog/2024/11/14/what-is-git-the-ultimate-guide-to-gits-role-and-functionality/>.
- [46] Leaflet doc, 2024. URL <https://leafletjs.com/reference.html>.
- [47] Open data handbook - cosa sono gli open data, 2025. URL <https://opendatahandbook.org/guide/en/what-is-open-data/>.
- [48] Tutorial css: introduzione al cascading style sheets, 2022. URL <https://www.ionos.it/digitalguide/siti-web/web-design/come-imparare-il-css-una-guida-semplice/>.

- [49] Xampp tutorial, 2025. URL <https://www.javatpoint.com/xampp>.