UNIVERSITÀ
degli STUDI
di CATANIA

Dipartimento di ingegneria elettrica, elettronica ed informatica

Ingegneria Informatica

Volchazard: interfaccia web per visualizzazione dati

Candidato: Leontini Alessandro
Relatore: Simone Palazzo

Anno Accademico 2022/2023

Sommario

/* bozza */ Nel campo in rapida evoluzione dell'informatica, la capacità di visualizzare e analizzare efficacemente i dati è fondamentale per prendere decisioni informate e sviluppare sistemi efficienti. I sistemi tradizionali di visualizzazione dei dati sono spesso caratterizzati da rigidità e scarsa riusabilità, risultando difficili da adattare a nuovi requisiti o da integrare con tecnologie moderne. Questa tesi affronta tali problematiche proponendo un'architettura di sistema modernizzata e riusabile per la visualizzazione dei dati, sfruttando principi contemporanei dell'ingegneria del software come la modularità, il design basato su componenti e la separazione delle responsabilità. Il progetto analizza i paradigmi e le tecnologie esistenti, ne identifica i limiti e introduce un framework flessibile, capace di presentare dati eterogenei su diverse piattaforme. Attraverso l'implementazione di componenti riusabili e interfacce dati standardizzate, la soluzione proposta facilita una rapida adattabilità ai cambiamenti nelle esigenze degli utenti e nei progressi tecnologici. L'efficacia del sistema viene dimostrata tramite casi di studio pratici e analisi delle performance, evidenziando miglioramenti in termini di manutenibilità, scalabilità ed esperienza utente. L'obiettivo di questo lavoro è offrire una solida base per futuri sviluppi in applicazioni data-centriche, promuovendo sostenibilità e innovazione nella progettazione dei sistemi di visualizzazione dei dati.

Abstract

In the rapidly evolving field of computer science, the ability to efficiently display and analyze data is critical for informed decision-making and effective system development. Traditional data display systems often suffer from rigidity and lack of reusability, making them difficult to adapt to new requirements or integrate with modern technologies. This thesis addresses these challenges by proposing a modernized, reusable system architecture for data display, leveraging contemporary software engineering principles such as modularity, component-based design, and separation of concerns. The project explores existing paradigms and technologies, identifies their limitations, and introduces a flexible framework capable of presenting diverse data types across various platforms. Through the implementation of reusable components and standardized data interfaces, the proposed solution facilitates rapid adaptation to changing user needs and technological advancements. The system's effectiveness is demonstrated through practical case studies and performance analyses, highlighting improvements in maintainability, scalability, and user experience. Ultimately, this work aims to contribute a robust foundation for future developments in data-centric applications, promoting sustainability and innovation in the design of data display systems.

Indice

Capitolo 1

Introduzione

1.1 Introduzione

Nel panorama attuale della scienza dei dati e dell'ingegneria, la capacità di raccogliere, analizzare e visualizzare grandi quantità di dati temporali rappresenta un requisito fondamentale per numerosi settori applicativi, dal monitoraggio ambientale alla gestione di infrastrutture critiche, fino all'analisi di fenomeni naturali complessi come le eruzioni vulcaniche. Tuttavia, la crescente complessità e quantità dei dati disponibili pone sfide significative sia dal punto di vista della progettazione delle interfacce utente sia, soprattutto, dal punto di vista delle performance dei sistemi di visualizzazione e interazione.

In questo contesto si colloca la presente tesi, il cui obiettivo principale è la progettazione e lo sviluppo di un'applicazione web per la visualizzazione interattiva di serie temporali di dati, massimizzando le performance in termini di reattività, scalabilità e fruibilità su diversi dispositivi. L'applicazione consente non solo l'e-

splorazione e il confronto di diversi stati dei dati, ma anche la loro esportazione e condivisione, funzionalità sempre più richieste in ambito scientifico e industriale.

1.1.1 Contesto e motivazioni

Negli ultimi anni, la digitalizzazione di strumenti di misura e sensori ha portato a una crescita esponenziale della quantità di dati raccolti in tempo reale. La necessità di disporre di strumenti efficaci per la visualizzazione e l'analisi di questi dati è diventata centrale in molteplici ambiti: ricerca scientifica, monitoraggio ambientale, controllo di processo, gestione di emergenze, e molti altri. In particolare, nel monitoraggio di fenomeni naturali come le eruzioni vulcaniche, la possibilità di accedere a dati storici, confrontarli con dati in tempo reale e navigare rapidamente tra differenti scenari rappresenta un valore aggiunto per ricercatori, tecnici e decisori.

Tuttavia, la gestione e la visualizzazione di dataset di grandi dimensioni in ambito web presenta numerose criticità: i tradizionali strumenti di visualizzazione possono soffrire di lentezze, blocchi o inefficienze, soprattutto su dispositivi mobili o in presenza di connessioni lente. Inoltre, la necessità di garantire reattività e fluidità nell'interazione, senza sacrificare la qualità della rappresentazione grafica e la precisione delle analisi, impone l'adozione di soluzioni tecnologiche avanzate.

1.1.2 Problema e rilevanza

Il problema affrontato in questa tesi riguarda dunque la progettazione e realizzazione di un sistema che permetta la visualizzazione interattiva di serie temporali di dati ad alte prestazioni, in grado di gestire dataset di grandi dimensioni e offrire funzionalità avanzate come il confronto di scenari, la navigazione storica e l'esportazione dei risultati. La rilevanza di questo problema è duplice: da un lato, risponde a esigenze concrete di numerosi settori applicativi; dall'altro, rappresenta una sfida tecnica di non banale risoluzione, soprattutto se si considera la necessità di garantire performance elevate su una vasta gamma di dispositivi e condizioni operative.

1.1.3 Difficoltà e limiti delle soluzioni esistenti

Le soluzioni esistenti per la visualizzazione di dati temporali in ambito web presentano diversi limiti. Molte librerie grafiche sono pensate per dataset di dimensioni moderate e non riescono a mantenere performance accettabili quando la quantità di dati cresce. Altre soluzioni, più orientate alla scalabilità, sacrificano spesso la qualità dell'interazione utente o risultano difficili da personalizzare per esigenze specifiche, come la gestione di stati multipli o la navigazione storica. Inoltre, la maggior parte degli strumenti disponibili non integra nativamente funzionalità di esportazione personalizzata o di sincronizzazione dello stato via URL, che risultano invece essenziali in contesti collaborativi o di reporting scientifico.

1.1.4 Obiettivi della tesi

L'obiettivo di questa tesi è duplice: da un lato, progettare un'architettura software in grado di massimizzare le performance nella visualizzazione e nell'interazione con serie temporali di dati; dall'altro, implementare un'applicazione concreta che dimostri l'efficacia delle soluzioni adottate. In particolare, si intendono perseguire i seguenti obiettivi principali:

- Garantire la reattività dell'interfaccia utente anche in presenza di grandi quantità di dati.
- Offrire funzionalità avanzate di confronto tra differenti stati/scenari dei dati.
- Permettere la navigazione storica e la condivisione dello stato dell'applicazione tramite URL.
- Integrare una funzione di esportazione che consenta di salvare e condividere le visualizzazioni.
- Assicurare la compatibilità e la responsività dell'applicazione su diversi dispositivi.

1.1.5 Soluzione proposta e vantaggi

La soluzione proposta si basa sull'utilizzo di tecnologie web moderne e performanti: Astro per il rendering server-side, SolidJS per la gestione reattiva dello stato

dell'interfaccia, e D3.js per la visualizzazione dinamica dei dati. L'adozione di questi strumenti consente di separare in modo efficiente la logica di business dalla presentazione grafica, ottimizzando così sia i tempi di caricamento sia la fluidità delle interazioni.

Un aspetto centrale della soluzione è la gestione ottimizzata dello stato tramite URL: ogni configurazione dell'interfaccia e ogni filtro applicato vengono automaticamente codificati nell'indirizzo della pagina, permettendo la navigazione, la condivisione e il ripristino esatto della visualizzazione. Inoltre, l'implementazione di una funzione di esportazione consente di generare file HTML standalone che includono la visualizzazione corrente, facilitando la diffusione e la conservazione dei risultati.

Dal punto di vista delle performance, sono state adottate numerose strategie di ottimizzazione, tra cui il caricamento asincrono dei dati, la virtualizzazione dei componenti grafici, e l'adozione di algoritmi efficienti per la gestione delle serie temporali. L'architettura modulare dell'applicazione consente inoltre una facile estendibilità e integrazione con nuove fonti dati o tipologie di visualizzazione.

1.1.6 Tecnologie e metodologie utilizzate

Le principali tecnologie adottate includono:

- **Astro**: framework per il rendering server-side e la generazione di siti web statici e dinamici.

- **SolidJS**: libreria JavaScript per la creazione di interfacce utente reattive ad alte prestazioni.
- **D3.js**: libreria per la manipolazione efficiente di dati e la generazione di visualizzazioni dinamiche e interattive.
- **MySQL**: sistema di gestione di database relazionali per l'archiviazione e la gestione dei dati.
- **TailwindCSS**: framework CSS per la creazione di interfacce moderne e responsive.

Dal punto di vista metodologico, il progetto ha seguito un approccio iterativo e incrementale: dopo una fase iniziale di analisi dei requisiti e delle soluzioni esistenti, sono stati prototipati i principali componenti e sono state eseguite sessioni di benchmarking per valutare le performance in diversi scenari. Le ottimizzazioni sono state introdotte progressivamente, sulla base dei risultati ottenuti e dei feedback raccolti.

1.1.7 Risultati ottenuti

L'applicazione sviluppata ha dimostrato di poter gestire in modo efficiente dataset di grandi dimensioni, mantenendo tempi di risposta ridotti e un'eccellente fluidità dell'interazione anche su dispositivi con risorse limitate. Le strategie di ottimizzazione adottate hanno permesso di ridurre sensibilmente i tempi di caricamento

e di aggiornamento delle visualizzazioni rispetto a soluzioni tradizionali. Le funzionalità di confronto tra stati, navigazione storica tramite URL e esportazione delle visualizzazioni sono state apprezzate sia da utenti tecnici sia da utenti meno esperti, confermando la validità dell'approccio scelto.

In sintesi, il lavoro svolto ha fornito un contributo concreto al tema della visualizzazione performante di serie temporali di dati in ambiente web, proponendo soluzioni tecniche e architetturali replicabili in contesti affini e facilmente estendibili a nuove esigenze applicative.

Capitolo 2

Stato dell'arte

Di seguito sono descritte le sezioni che, in generale, dovrebbero comporre una tesi di laurea. La suddivisione presentata non è strettamente vincolante e dovrebbe essere adattata in base alle esigenze e alle caratteristiche della tesi trattata.

2.1 Stato dell'arte

Nel contesto della visualizzazione e analisi di serie temporali di dati scientifici e ingegneristici, esistono numerose soluzioni e piattaforme che affrontano, con diversi approcci, le criticità legate all'efficienza, alla scalabilità e all'interattività. In questo capitolo vengono analizzate le principali soluzioni esistenti, valutandone le caratteristiche, i punti di forza e le limitazioni, con particolare attenzione alle differenze rispetto alla soluzione proposta in questa tesi.

2.1.1 Grafana

Grafana è una delle piattaforme open source più diffuse per la visualizzazione e il monitoraggio di serie temporali di dati. Offre un'interfaccia utente altamente personalizzabile, il supporto a numerosi database (tra cui Prometheus, InfluxDB, MySQL) e la possibilità di creare dashboard interattive. Grafana è particolarmente apprezzata per la sua modularità e per l'ampia disponibilità di plugin.

Vantaggi:

- Interfaccia user-friendly e altamente personalizzabile.
- Ampio supporto a fonti dati eterogenee.
- Plugin per integrazione con sistemi di alerting e notifiche.
- Attiva comunità open source.

Svantaggi:

- Alcune funzionalità avanzate richiedono la versione enterprise.
- La gestione di dataset molto grandi può risultare onerosa in termini di risorse.
- Lato client non sempre ottimizzato per dispositivi mobili.
- La condivisione dello stato tramite URL non è sempre completa e trasparente.

Differenze rispetto alla soluzione proposta: La soluzione proposta, rispetto a Grafana, si concentra su una gestione più efficiente dello stato tramite URL, sull'esportazione avanzata delle visualizzazioni e su una maggiore ottimizzazione delle prestazioni lato client, soprattutto in ambiente mobile.

2.1.2 Kibana

Kibana è un tool open source principalmente utilizzato per la visualizzazione di dati provenienti da Elasticsearch, ma offre anche funzionalità avanzate di esplorazione e analisi di dati temporali. Kibana consente la creazione di dashboard e report personalizzati, con un focus particolare su dati di log e metriche di sistema.

Vantaggi:

- Integrazione nativa con Elastic Stack.
- Potenti funzionalità di ricerca, filtro e aggregazione.
- Dashboard interattive e personalizzabili.

Svantaggi:

- Dipendenza da Elasticsearch come backend dati.
- Scalabilità e performance dipendono dalla configurazione del cluster.
- Curva di apprendimento relativamente elevata per utenti non tecnici.

Differenze rispetto alla soluzione proposta: La soluzione proposta offre maggiore flessibilità nella scelta della sorgente dati e una più semplice esportazione dei risultati, oltre a una maggiore leggerezza e semplicità di installazione.

2.1.3 Plotly Dash

Plotly Dash è un framework Python per la creazione di applicazioni web interattive di visualizzazione dati. Consente la realizzazione di dashboard dinamiche, con grafici di elevata qualità e interattività avanzata.

Vantaggi:

- Facilità di utilizzo per chi ha familiarità con Python.
- Ampia gamma di visualizzazioni e grafici interattivi.
- Possibilità di integrare facilmente altri moduli Python per analisi dati.

Svantaggi:

- Performance limitate su dataset di grandi dimensioni.
- Scalabilità server-side non sempre ottimale senza configurazioni avanzate.
- Lato client non sempre reattivo per interazioni complesse.

Differenze rispetto alla soluzione proposta: La soluzione proposta, essendo sviluppata interamente in ambiente JavaScript/TypeScript e con tecnolo-

gie moderne come SolidJS e Astro, garantisce migliore reattività lato client e un'integrazione più immediata con ecosistemi web già esistenti.

2.1.4 Highcharts Cloud

Highcharts Cloud è una piattaforma SaaS che permette di creare, personalizzare e condividere grafici interattivi direttamente dal browser.

Vantaggi:

- Interfaccia intuitiva e immediata.
- Ampia varietà di grafici disponibili.
- Possibilità di esportazione in diversi formati.

Svantaggi:

- Funzionalità avanzate disponibili solo a pagamento.
- Limitazioni nella personalizzazione dei flussi dati e delle interazioni.
- Non adatto a integrazioni personalizzate o workflow complessi.

Differenze rispetto alla soluzione proposta: La piattaforma sviluppata in questa tesi consente una piena integrazione con flussi dati personalizzati e la gestione avanzata dello stato tramite URL, offrendo maggiore flessibilità e possibilità di estensione.

2.1.5 Altre soluzioni open source e librerie JavaScript

Esistono numerose librerie JavaScript per la visualizzazione di serie temporali, tra cui D3.js, Chart.js, e ECharts. Queste librerie forniscono strumenti potenti per la creazione di grafici, ma richiedono spesso sviluppi ad hoc per integrare funzionalità avanzate come la gestione dello stato, il confronto tra scenari, o l'esportazione delle visualizzazioni.

Vantaggi:

- Estrema flessibilità e possibilità di personalizzazione.
- Ampia comunità di supporto.
- Elevate performance se utilizzate correttamente.

Svantaggi:

- Necessità di competenze avanzate di sviluppo front-end.
- Mancanza di funzionalità pronte all'uso per la gestione avanzata dello stato o l'esportazione.
- Maggiore effort di integrazione con sistemi backend.

Differenze rispetto alla soluzione proposta: La soluzione proposta integra nativamente funzionalità di gestione avanzata dello stato, esportazione e confronto, riducendo così il tempo e l'effort richiesti per ottenere un prodotto finale pronto all'uso.

2.2 Sintesi e confronto

Dall'analisi delle soluzioni esistenti emerge che, sebbene siano disponibili strumenti potenti e flessibili per la visualizzazione di dati temporali, nessuna delle piattaforme considerate offre contemporaneamente:

- prestazioni elevate anche su dataset di grandi dimensioni;
- gestione avanzata dello stato e della navigazione tramite URL;
- funzionalità di confronto tra stati multipli;
- esportazione facile e personalizzata delle visualizzazioni;
- massima compatibilità e performance su dispositivi mobili e desktop.

La soluzione proposta in questa tesi si colloca quindi come un'alternativa innovativa, in grado di coniugare le migliori pratiche di sviluppo web moderno con funzionalità avanzate e performance ottimizzate per le esigenze della ricerca scientifica e dell'industria.

Capitolo 3

Tecnologie

Di seguito sono descritte le sezioni che, in generale, dovrebbero comporre una tesi di laurea. La suddivisione presentata non è strettamente vincolante e dovrebbe essere adattata in base alle esigenze e alle caratteristiche della tesi trattata.

3.1 Tecnologie (solo DBWP)

In questo capitolo vengono descritte le principali tecnologie adottate per la realizzazione del progetto, analizzandone le caratteristiche, i vantaggi, i limiti e le motivazioni che hanno portato alla loro scelta. L'integrazione efficace di tali strumenti ha permesso di raggiungere gli obiettivi di performance, scalabilità e usabilità richiesti dal contesto applicativo.

3.1.1 Astro

Astro è un moderno framework per la creazione di siti web statici e dinamici, ottimizzato per il rendering lato server (SSR) e la generazione di pagine leggere. Grazie

alla sua architettura "island-based", Astro consente di caricare solo il codice strettamente necessario per ogni componente della pagina, riducendo drasticamente il payload e migliorando i tempi di caricamento. Astro supporta l'integrazione di diversi framework JavaScript (come React, SolidJS, Vue) e permette la generazione di siti sia statici che dinamici, risultando particolarmente adatto per applicazioni ad alte prestazioni.

Vantaggi:

- Rendering server-side efficiente.
- Ottimizzazione automatica delle risorse.
- Supporto a molteplici framework UI.
- Riduzione significativa del JavaScript lato client.

Svantaggi:

- Ecosistema ancora giovane.
- Minore disponibilità di plugin rispetto a framework più maturi.

3.1.2 SolidJS

SolidJS è una libreria JavaScript per la costruzione di interfacce utente reattive, focalizzata sulla massimizzazione delle performance tramite un sistema di reattività fine-grained e la generazione di codice minimale. SolidJS non utilizza un

virtual DOM, ma aggiorna il DOM reale solo dove strettamente necessario, garantendo performance superiori rispetto ad alternative più diffuse come React o Vue, soprattutto in presenza di dataset di grandi dimensioni.

Vantaggi:

- Aggiornamenti estremamente rapidi e granulari.
- Consumo di memoria ridotto.
- Facilità di integrazione con Astro.

Svantaggi:

- Comunità e documentazione in crescita ma ancora limitate.
- Alcune librerie di terze parti non ancora pienamente compatibili.

3.1.3 D3.js

D3.js (Data-Driven Documents) è la libreria di riferimento per la manipolazione di dati e la generazione di visualizzazioni dinamiche e interattive in ambiente web. D3 permette di mappare dati complessi su elementi grafici SVG, Canvas o HTML, abilitando la creazione di grafici altamente personalizzati e interattivi. Nel progetto è stato utilizzato per la costruzione dei grafici temporali, la gestione dello zoom e della panoramica, e l'aggiornamento dinamico delle visualizzazioni.

Vantaggi:

- Estrema flessibilità e potenza espressiva.
- Supporto a tutte le tipologie di dati e grafici.
- Ottimizzazioni per grandi moli di dati.

Svantaggi:

- Curva di apprendimento elevata.
- Richiede una buona conoscenza di JavaScript e del DOM.

3.1.4 MySQL

MySQL è uno dei database relazionali open source più diffusi e utilizzati al mondo. Offre elevate performance in termini di gestione delle query e scalabilità, risultando adatto sia per applicazioni di piccole dimensioni che per sistemi enterprise. Nel progetto viene utilizzato per l'archiviazione e la gestione delle serie temporali di dati, sfruttando la sua affidabilità, la facilità di integrazione e il supporto a query complesse.

Vantaggi:

- Performance elevate per la gestione di dati strutturati.
- Ampio supporto comunitario e documentazione completa.
- Facilità di backup e replicazione.

Svantaggi:

- Minor efficienza per l'analisi di dati non strutturati.
- Scalabilità orizzontale limitata rispetto a soluzioni NoSQL.

3.1.5 TailwindCSS

TailwindCSS è un framework CSS utility-first che consente di costruire interfacce moderne e responsive in modo rapido e modulare. Le classi utility permettono di definire direttamente nello stile degli elementi le proprietà di layout, colore, tipografia e spaziatura, semplificando la manutenzione e l'evoluzione dell'interfaccia grafica.

Vantaggi:

- Rapidità di sviluppo delle interfacce.
- Elevata personalizzazione e modularità.
- Ottimizzazione automatica del CSS finale.

Svantaggi:

- Possibile aumento della verbosità nel markup.
- Richiede una fase di apprendimento per l'approccio utility-first.

3.1.6 Node.js e Vite

Node.js rappresenta la piattaforma di esecuzione per il backend, mentre Vite è lo strumento di build scelto per ottimizzare i tempi di sviluppo e caricamento in ambiente locale. La combinazione di questi strumenti garantisce tempi di risposta ridotti, hot-reloading e facilità di integrazione con le moderne tecnologie frontend.

Vantaggi:

- Esecuzione asincrona e non bloccante (Node.js).
- Build e reload ultraveloci (Vite).
- Ampia compatibilità con ecosistemi JavaScript e TypeScript.

Svantaggi:

- Necessità di aggiornamenti frequenti per mantenere la sicurezza.
- Dipendenza da ecosistemi open source in rapida evoluzione.

3.1.7 Architettura modulare e best practice adottate

L'architettura del progetto segue principi di modularità, separazione delle responsabilità e riusabilità del codice. I principali moduli (visualizzazione, gestione dati, interfaccia utente, API, export) sono stati progettati in modo indipendente, agevolando la manutenzione e l'estendibilità futura. Sono state adottate best practice di sviluppo come la gestione centralizzata dello stato, l'utilizzo di context provider, la scrittura tipizzata in TypeScript e la documentazione del codice.

3.1.8 Integrazione e deploy

Il processo di integrazione e deploy è stato automatizzato tramite script di build e configurazioni dedicate, garantendo la possibilità di eseguire facilmente il progetto sia in ambiente di sviluppo che in produzione. Particolare attenzione è stata posta all'ottimizzazione delle performance tramite tecniche di code splitting, lazy loading e compressione delle risorse statiche.

3.1.9 Considerazioni finali sulle tecnologie

La scelta delle tecnologie descritte si è rivelata vincente per il raggiungimento degli obiettivi di progetto: reattività dell'interfaccia, efficienza nella gestione dei dati, scalabilità e facilità di manutenzione. L'integrazione di strumenti moderni e performanti ha permesso di ottenere un prodotto finale robusto, flessibile e facilmente estendibile, pronto per affrontare le sfide poste dalla crescente complessità dei dati e delle esigenze degli utenti.

Capitolo 4

Conclusioni

Di seguito sono descritte le sezioni che, in generale, dovrebbero comporre una tesi di laurea. La suddivisione presentata non è strettamente vincolante e dovrebbe essere adattata in base alle esigenze e alle caratteristiche della tesi trattata.

4.1 Conclusioni

L'obiettivo di questa tesi è stato quello di progettare e sviluppare un'applicazione web performante per la visualizzazione interattiva, il confronto e l'esportazione di serie temporali di dati, in particolare in ambito scientifico e ingegneristico. La soluzione proposta si fonda su un'architettura moderna che integra Astro per il rendering server-side, SolidJS per la gestione reattiva dell'interfaccia utente e D3.js per la visualizzazione dinamica dei dati, con MySQL come backend per la gestione efficiente di serie temporali anche di grandi dimensioni.

L'applicazione realizzata consente di esplorare e confrontare diversi scenari di dati, navigare in modo storico tramite la gestione avanzata dello stato via URL,

ed esportare facilmente le visualizzazioni per analisi e reporting offline. L'adozione di un'architettura modulare e di strategie di ottimizzazione mirate ha permesso di raggiungere elevati livelli di performance e reattività, anche su dispositivi mobili o in condizioni di connettività limitata.

I risultati ottenuti mostrano come la soluzione sia in grado di gestire dataset di grandi dimensioni mantenendo tempi di risposta ridotti e fluidità nelle interazioni, confermando la validità delle scelte tecnologiche e architetturali adottate. Le funzionalità di confronto tra stati, esportazione e navigazione storica sono risultate particolarmente apprezzate dagli utenti durante le fasi di test e validazione.

Nonostante i risultati positivi, la soluzione proposta presenta alcune limitazioni. In particolare, la gestione di dataset estremamente voluminosi potrebbe richiedere ulteriori ottimizzazioni lato backend, come l'adozione di tecniche di indicizzazione avanzata, caching o l'utilizzo di database NoSQL per particolari casi d'uso. Inoltre, l'integrazione con fonti dati eterogenee o in tempo reale rappresenta una possibile evoluzione futura, così come lo sviluppo di ulteriori componenti di analisi avanzata e reporting automatico.

Il codice sorgente dell'applicazione è disponibile pubblicamente su GitHub all'indirizzo: `https://github.com/aleleo1/project`

Eventuali risultati di ricerca o pubblicazioni derivanti dal presente lavoro saranno menzionati e resi disponibili nella piattaforma open access dell'ateneo.

In conclusione, questa tesi ha fornito un contributo concreto al tema della visualizzazione performante di dati scientifici in ambiente web, proponendo una soluzione moderna, efficiente e facilmente estendibile, che potrà fungere da base per sviluppi futuri nel campo della data visualization e dell'analisi interattiva dei dati.

Esempio bibliografia

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[1]