

ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Triennale

Titolo

Relatore
prof. Paolo Bellavista

Laureando
Davide Crociati

Ottobre 2023

A voi

Sommario

Piacere, sommario.

Indice

Elenco delle figure	v
Elenco delle tabelle	vi
Elenco dei listati	vii
1 Introduzione	1
1.1 Contestualizzazione	1
1.1.1 Evoluzione delle applicazioni web	1
1.1.2 Importanza dell'ottimizzazione	2
1.2 Motivazioni e Obiettivi	3
1.2.1 Analisi Comparativa delle Tecnologie	4
1.2.2 Valutazione dell'Impatto di Wasm	4
1.3 Trend di evoluzione del web	4
1.3.1 Paradigmi di Sviluppo Moderni	4
1.3.2 Esperienze Utente Migliorate ?	4
1.4 Struttura della tesi	4
2 Capitolo 2	6
2.1 Prototipo	6
3 Capitolo 3	7
3.1 Prototipo	7
Riferimenti bibliografici	8

Elenco delle figure

1.1	Differenza nel tipo di richieste tra SPA e MPA.	2
1.2	Il campo dell'elaborazione digitale di immagini è molto ampio e sarebbe possibile integrare anche operazioni avanzate (edge detection, pattern recognition etc.) che richiederebbero risorse computazionali ancora maggiori, ma per lo scopo di questa tesi ci si limiterà a elaborazioni più semplici.	3
1.3	Rust + Wasm	4
1.4	Node.js	4
1.5	This is not a figure. It's a caption.	5

Elenco delle tabelle

Elenco dei listati

1.1	I directly included a portion of a file	4
1.2	Some code in another language than the default one	4

Capitolo 1

Introduzione

1.1 Contestualizzazione

1.1.1 Evoluzione delle applicazioni web

Inizialmente le applicazioni web erano costituite da semplici pagine statiche contenenti testo e immagini. Col passare degli anni, grazie all'adozione di JavaScript e di librerie e framework correlati, hanno progressivamente acquisito un carattere più dinamico, con l'introduzione di livelli crescenti di interattività. Un cambiamento significativo in tal senso, è avvenuto con l'avvento delle **Single Page Application (SPA)** e di **AJAX**. Tale combinazione, ha infatti introdotto un nuovo paradigma di sviluppo, in cui l'intera applicazione viene caricata una sola volta e le successive interazioni con l'utente avvengono grazie al caricamento dinamico di contenuti e dati provenienti da un web server, eliminando così l'attesa nel caricamento di una nuova pagina e favorendo l'esperienza d'uso.

Parallelamente, la complessità delle funzionalità offerte è cresciuta in modo esponenziale, spaziando da applicazioni di grafica 3D, a simulatori, ad applicazioni di modifica di documenti, immagini e video. Oggi, è sempre più comune incontrare siti web in grado di gestire complesse operazioni in tempi rapidi, assicurando così quell'interattività alla quale ormai siamo abituati. Tuttavia questo progresso è stato accompagnato da un aumento del numero di richieste effettuate in rete e dall'utilizzo intensivo di risorse computazionali, sia lato cliente, che lato servitore.

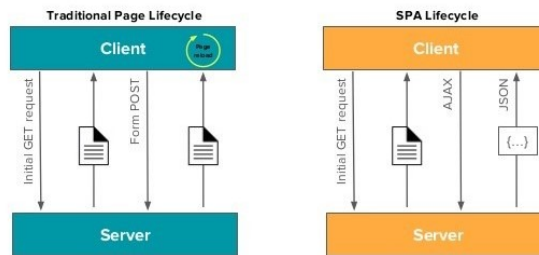


Figura 1.1: Differenza nel tipo di richieste tra SPA e MPA.

1.1.2 Importanza dell'ottimizzazione

Ad oggi, l'ottimizzazione delle prestazioni è quindi diventata un aspetto cruciale nello sviluppo di applicazioni web. Gli utenti si aspettano interazioni con bassa latenza, caricamenti rapidi e risposte immediate. Questa esigenza mette in risalto l'importanza di bilanciare l'aggiunta di funzionalità sofisticate, con l'offerta di una *User Experience* ottimale.

I tempi di caricamento prolungati possono portare a un alto tasso di abbandono delle pagine, riducendo l'opportunità di coinvolgere nuovi utenti. Inoltre, con l'aumentare dell'utilizzo di dispositivi mobili e di conseguenza, di connessioni instabili, l'ottimizzazione diventa ancor più critica per assicurare un'esperienza coerente su diverse piattaforme e condizioni di rete. Tutto ciò non riguarda solo il lato client, ma coinvolge anche il lato server. Un carico eccessivo sui server può influire negativamente sulla scalabilità, causando ritardi nelle risposte e possibili interruzioni del servizio. L'ottimizzazione deve quindi coinvolgere tutti gli aspetti dell'architettura delle applicazioni web.

Nell'implementare ottimizzazioni, sono nati varie soluzioni interessanti. Ad esempio, per gestire task che svolgono molte operazioni di Input/Output si è distinto **Node.js**, mentre per quanto riguarda l'esecuzione di attività che sfruttano molto la CPU è emerso **WebAssembly(Wasm)**, in sinergia con l'interfaccia di sistema **WebAssembly System Interface (WASI)**.

1.2 Motivazioni e Obiettivi

La crescente complessità delle applicazioni web e l'esigenza di offrire agli utenti esperienze interattive sempre più coinvolgenti hanno portato l'ambito dello sviluppo web a una svolta significativa. Le aspettative degli utenti si sono evolute verso applicazioni che offrano prestazioni reattive, interattività immediata e funzionalità avanzate. È proprio questo confronto a essere alla base delle motivazioni che hanno guidato la scelta del tema di questa tesi di laurea. In particolare, la presente ricerca si propone di esplorare in profondità il complesso equilibrio tra l'implementazione di funzionalità sofisticate (principalmente **CPU-Intensive**) e l'ottimizzazione delle prestazioni all'interno delle applicazioni web. Il fulcro di questa indagine sarà una comparazione dettagliata tra due approcci di sviluppo distinti per un'applicazione che consenta **l'elaborazione server-side di immagini** caricate da un utente.

Tale tipologia di applicazione, si sposa bene per lo scopo finale della tesi: valutare come l'utilizzo del linguaggio di programmazione Rust con l'integrazione di WebAssembly (Wasm/WASI), e l'approccio basato su Node.js, affrontino il problema di un'applicazione web che svolga operazioni dall'alto costo computazionale, fornendo un'analisi approfondita delle performance e delle prestazioni riscontrate. Si intende esplorare le opportunità offerte da tecnologie quali WebAssembly e Node.js nell'ottica di un'ottimizzazione delle prestazioni. Questa ricerca mira a comprenderne i benefici specifici, individuando le situazioni in cui uno dei due approcci può risultare più vantaggioso in termini di efficienza computazionale e reattività.



Figura 1.2: Il campo dell'elaborazione digitale di immagini è molto ampio e sarebbe possibile integrare anche operazioni avanzate (edge detection, pattern recognition etc.) che richiederebbero risorse computazionali ancora maggiori, ma per lo scopo di questa tesi ci si limiterà a elaborazioni più semplici.

1.2.1 Analisi Comparativa delle Tecnologie

Si partirà con un analisi comparativa delle due tecnologie, presentando i vantaggi e gli svantaggi teorici di entrambe.



Figura 1.3: Rust + Wasm



Figura 1.4: Node.js

1.2.2 Valutazione dell'Impatto di Wasm

1.3 Trend di evoluzione del web

1.3.1 Paradigmi di Sviluppo Moderni

1.3.2 Esperienze Utente Migliorate ?

1.4 Struttura della tesi

This is a reference to a chapter 1. This is a reference to a figure 1.5. This is a reference to some code 1.1. This is a citation [2].

```
1 import suca
2
3 print "hello_world!"
```

Listato 1.1: I directly included a portion of a file

```
1 public void prepare(AClass foo) {
2     AnotherClass bar = new AnotherClass(foo)
3 }
```

Listato 1.2: Some code in another language than the default one



Figura 1.5: This is not a figure. It's a caption.

Capitolo 2

Capitolo 2

2.1 Prototipo

Capitolo 3

Capitolo 3

3.1 Prototipo

Ringraziamenti

Grazie a Grazia, Graziella e la sorella.

Bibliografia

- [1] Carlino Cane. *No one cited me*. A cura di Anche Leggoo. first edition. O'Reilly, 2009.
URL: <http://bar.com/>.
- [2] Darth Vader e Neo Cortex. «Hell yeah! I'm a super important paper!» In: *Stampo Solo* 51 (2008), pp. 107–113. DOI: [10.1145/1327452.1327492](https://doi.org/10.1145/1327452.1327492). URL: <http://www.foo.com>.