Università degli Studi di Firenze

Scuola di Ingegneria - Dipartimento di Ingegneria dell'Informazione

---

Tesi di Laurea Triennale in Ingegneria Informatica

Microfrontends

Candidato
Matteo Bavecchi

Relatori
Prof. Romano Fantacci
Prof. Giuseppe Pecorella

Correlatore
Dott. Andrea Rizzo

Anno Accademico 2020/2021

Indice

ringraziamenti

Grazie a tutti

Introduzione

Organizzare il lavoro per lo sviluppo di progetti web di grandi dimensioni non è per niente banale, e può seguire diverse logiche di suddivisione di responsabilità tra le parti al quale contribuiscono. L'approccio più diffuso è quello di suddividere le persone per competenze, creando team che mettono in comune figure con abilità dello stesso ambito, che contribuiscono ad una parte del progetto complessivo. Ad esempio in un sito e-commerce possiamo trovare un team che si occupa della parte frontend, uno che cura i servizi di pagamento e uno che segue la parte backend. Quando il progetto aumenta di complessità, si sente la necessità di suddividere il lavoro in sotto-progetti, e l'approccio orizzontale potrebbe non essere la scelta migliore, in quanto rallenta l'introduzione di nuove funzionalità.

Possiamo allora pensare di assegnare ad ogni team una parte del progetto, i quali dovranno portarlo al termine interamente. Ogni team avrà bisogno quindi di competenze eterogenee al loro interno.

Capitolo 1

Microfrontend

L'obiettivo della tecnologia microfrontend è quello di superare l'approccio monolitico, che vede lo sviluppo di applicazioni web suddiviso in due teams: backend e frontend.

Si fa questo vedendo un'applicazione web come un insieme di elementi, chiamati fragment o microfrontend, molto disaccoppiati tra di loro e con la più bassa granuralità, ovvero con la funzionalità più minimale possibile.

Ogni microfrontend viene sviluppato da un team, che potrà lavorare con autonomia. Essendo i singoli microfrontend autonomi, questi possono funzionare anche se estratti dalla applicazione web che li contiene, e il malfunzionamento di un singolo microfrontend non compromette la stabilità degli altri.

Vantaggi

- **Ottimizzare sviluppo di funzionalità:** Nell'approccio orizzontale quando si vuole sviluppare una nuova funzionalità ci si deve accordare tra vari team e fare molti incontri per discutere. Con i microfrontend, tutte le persone coinvolte nell'implementazione di una nuo-

va funzionalità sono nello stesso team, e quindi è tutto più veloce ed efficiente.

- **Abbandono del frontend monolitico:** La maggior parte delle acrchitetture oggi non hanno il concetto di scalablità nel frontend, ma per questo hanno un concetto monolitico. Si può pensare di dividere il backend e il frontend, oppure di rendere il backend l'insieme di tanti microservizi, ma il frontend rimane unico. Con microfrontend le applicazioni, incluse il frontend, si dividono in sistemi verticali più piccoli. Ogni team controlla la sua piccola parte di frontend. Questo porta diversi benefici, come l'isolamento dei rischi di fallimento, e maggiore predicibilità, in quanto si ha pezzi più piccoli di sistema che non condividono molti stati con il resto dell'applicazione.

- **Abilità nel cambiare:** Strumenti di sviluppo e framework evolvono continuamente. E' essenziale per un team essere in grado di adottare una nuova tecnologia quando questo ha senso. Ci sono esempi di aziende che hanno avuto bisogno di adottare strumenti diversi che hanno dovuto fare delle transizioni epocali, alcune durate diversi anni. Come il caso di Github, che ha impiegato molto tempo per eliminare le dipendenze da JQuery dal loro codice. Con l'approccio microfrontend questi cambiamenti sono più rapidi e possono essere fatti modularmente.

- **Indipendenza:** I frammenti di ogni team sono autonomi, ovvero non hanno dipendenze condivise a runtime. Questo permette ad ogni team di introdurre funzionalità senza consultare altri.

  L'indipendenza però porta sicuramente a costi aggiuntivi. Si potrebbe pensare che sia più semplice quindi fare un unica applicazione di grandi dimensioni di cui ogni team è responsabile di una parte diversa. Il

problema sta nel fatto che la comunicazione tra i vari team è costosa e porta molti ritardi. Quindi si preferisce addirittura la ridondanza a favore di più autonomia e velocità di implementazioni.

Svantaggi

- **Ridondanza:** In informatica si è addestrati a ridurre al minimo ridondanze, che si parli di normalizzazione dei dati nei database o di estrarre parti di codici condivisi per farne una funzione. Anche nel caso dello sviluppo frontend ci sono degli episodi nei quali la ridondanza può essere molto costosa: come ad esempio quando viene trovato un bug in una libreria e questo viene risolto da un team, questo poi dovrà essere comunicato agli altri e questi dovranno provvedere a risolverlo autonomamente. Oppure quando si rende un processo più veloce, anche in questo caso va comunicato agli altri team e questi dovranno apprendere la scoperta. Si adotta quindi un approccio microfrontend quando i costi associati a ridondanze sono inferiori agli impatti negativi a uno sviluppo frontend monolitico, che porta a forti dipendenze tra team.

- **Inconsistenza:** I database usati nell'applicazione devono essere a disposizione di tutti i team. Quindi quello che si fa di solito è replicare il database per tutti i team. Tutte queste copie vengono aggiornate regolarmente per mantenere la consistenza e la coerenza. Ma ci potrebbero essere dei ritardi. Di solito si parla di millisecondi o al massimo secondi, in casi peggiori invece l'attesa potrebbe essere più lunga. Questo è un altro compromesso che privilegia la robustezza alla coerenza garantita.

- **Eterogeneità:** Potrebbe essere controverso avere la libertà di utilizzare tecnologie diverse tra i vati team. Ovviamente se ogni team usa

una diversa tecnologia lo scambio di pareri o di competenze tra team diventa più difficile Ma questo non deve essere un obbligo. Da un elenco di tecnologie collaudate un team si sentirà libero di scegliere quella più accettabile.

Capitolo 2

Composizione di microfrontends

Eccoci qui: [?]

Capitolo 3

Conclusioni

...