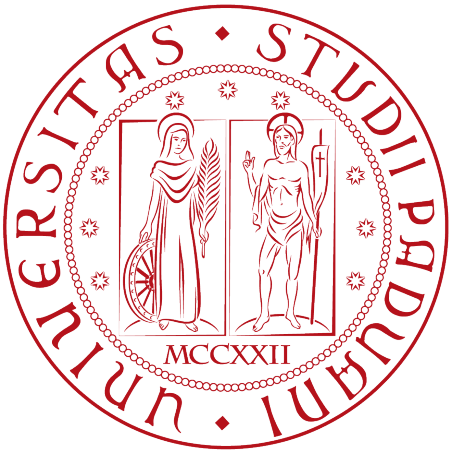
Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea in Informatica



Editor visuale per la manipolazione di template HTML

Tesi di laurea triennale

Relatore

Prof. Claudio Enrico Palazzi

Laureando

Daniele Marin

Anno Accademico 2016-2017

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di trecentoventi ore, dal laureando Daniele Marin presso l'azienda Zucchetti S.p.a.. L'obiettivo di tale attività di stage è l'analisi di varie librerie Javascript per la realizzazione di template HTML, al fine di poter realizzare un editor grafico che permetta la selezione e la modifica dei template per un loro sucessivo inserimento all'interno di pagine HTML. Inoltre è stato effettuato uno studio sul comportamento dei template in ambito responsive, sulla possibilità di inserire plug-in jQuery all'interno dei template e su di un metodo di caricamento delle librerie controllato in modo di non avere più istanze della stessa libreria se utilizzata da diversi template.

Ringraziamenti

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei frammenti di codice

Capitolo 1

Introduzione

1.1 L'azienda

Figura 1.1: Logo di Zucchetti S.p.a.

La Zucchetti S.p.a. è una software house con sede a Lodi, che si occupa di soluzioni complete per le aziende, professionisti(commercialisti, consulenti del lavoro, avvocati, curatori fallimentari, notai ecc.) e pubbliche amministrazioni(Comuni, Province, Regioni, Ministeri, società pubbliche ecc.).
Il gruppo Zucchetti è la prima azzienda italiana in Europa con oltre 3300 addetti, 1100 partner e oltre 105000 clienti.
Le soluzioni principali proposte dall'azienda sono :

- Softwere: gestionali, per la sicurezza sul lavoro, analisi business ecc.
- Hardware: per la rilevazione presenze, controllo accessi e controllo produzione.
- Servizi: di outsourcing, cloud computing e data center.

1.1.1 Portal Studio

Lo stage si è svolto nella sede distaccata di Padova che si occupa di ricerca e sviluppo. Tra i software che vengono sviluppati in questa sede è presente Portal Studio che consiste in una WEB application per la creazione di siti web.
L'applicazione offre all'utente un set completo di strumenti per la creazione di pagine web, permette la creazione e modifica in modo grafico della struttura HTML, la gestione degli stili tramite editor grafico per il CSS ed inoltre permette di gestire i dati provenienti da diversi tipi di database, il loro filtraggio e il binding con varie strutture HTML come liste e tabelle.
Il software risulta essere molto maturo e oltre alle funzionalità sopracitate permette

anche la creazione di portlet e pagelet e altri elementi riutilizzabili e la gestione di risorse come dati in formato JSON.

1.2 Il progetto

Il progetto proposto dall'azienda per lo stage, nasce dal desiderio di aggiungere all'applicazione Portal Studio una nuova funzionalità che consiste nell'offrire all'utente la possibilità di inserire nelle proprie pagine HTML dei template già pronti e selezionabili da un insieme prestabilito.
Questo desiderio ha portato l'azienda ad interessarsi ai template engine come Mustache.js, HandlebarJS ecc.
Lo stage si divide in due parti.
La prima consisteva nello studio dei template e degli aspetti ad essi correlati, la seconda nella realizzazione di un editor che ne permettesse la visualizzazione e la modifica.

1.2.1 Prima parte

La prima parte del progetto inizia con la realizzazione di qualche template prototipo, utile sia per studiare le possibilità della libreria scelta sia per avere un insieme di template da inserire nell'editor che è stato realizzato in seguito.
Durante questa parte del progetto l'attenzione è stata rivolta alla possibilità di realizzare template statici, dinamici, template come composizione di altri template (es. lista di contatti) e template che utilizzano SVG.
In seguito alla realizzazione dei template prototipo è stato effettuato uno studio sulla possibilità di rendere i template responsive cioè permetterne la visualizzazione sia su dispositivi desktop che mobile.
La prima parte si è conclusa con uno studio sulla possibilità di realizzare template che contenessero al loro interno plug-in JQuery e sulla gestione del caricamento delle librerie necessarie al funzionamento dei template all'interno della pagina HTML.

1.2.2 Seconda parte

La seconda parte del progetto consisteva nella realizzazione di un editor grafico che permette all'utente la selezione di un template da una lista prestabilita. In seguito alla selezione del template desiderato quest'ultimo verrà visualizzato in un box dedicato e tramite un editor, che viene costruito sulla base dei dati editabili del template (i dati sono contenuti in un oggetto JSON che fa parte del template), è possibile vedere il comportamento del template durante la modifica dei suoi dati.
Per determinati template, come quelli considerati composti, l'editor deve dare la possibilità di visualizzare direttamente l'oggetto JSON contenente i dati e permetterne la modifica.
Non essendo presente una struttura di beck-end proposta dall'applicazione Portal Studio, perché ancora in fase di valutazione, l'editor è stato sviluppato separatamente, l'insieme dei template consiste in un gerarchia di directory contenenti i vari elementi che compongono i template (file HTML, JSON, immagini e librerie) suddivise per categoria.
Il caricamento delle risorse viene eseguito tramite chiamate http-get request, non essendo presenti delle API fornite dall'azienda.

Capitolo 2

Librerie analizzate

In questo capitolo vengono messe a confronto varie librerie **JavaScript** che permettono la realizzazione di template HTML, ne vengono analizzati i pregi e i difetti per arrivare a descrivere i motivi che hanno portato all scelta della libreria utilizzata nel progetto.

2.1 Considerazioni generali

Negli ultimi anni sono nate molte librerie che permettono la creazione di template HTML che hanno portato notevoli vantaggi agli sviluppatori, offrendo loro un nuovo strumento che permette di creare modelli HTML per la rappresentazione dei dati e riutilizzarli all'interno di pagine differenti con una considerevole diminuzione del codice JavaScript e HTML che normalmente viene utilizzato per la modifica de DOM. Queste librerie si sono evolute velocemente fino ad arrivare a permettere agli sviluppatori di creare intere User interface per applicazioni web, creare componenti riutilizzabili ed in qualche caso offrire funzionalità avanzate come il two-way binding.

2.1.1 I template con sintassi mustache

Le librerie studiate durante lo stage utilizzano tutte questa particolare sintassi, che permette di rappresentare variabili, sezioni, parziali ed altri elementi utili alla creazione del template, tramite l'inserimento di **tag**.
Questi particolari **tag** sono caratterizzati dall'utilizzo delle parentesi graffe come delimitatori e questo è il motivo per cui vengono definiti mustaches (baffi in inglese).
I tag si presentano nella forma "{{I P}}" dove I è un simbolo o una stringa ed identifica il tipo di tag, mentre P è un parametro o una chiave appartenente all'oggetto JSON correlato al template.
Per l'inserimento di variabili o parziali il **tag** è singolo, mentre per l'inserimento di sezioni, controlli del tipo not-exist ed altri sono presenti un **tag** di apertura ed uno di chiusura.
Per capire meglio il funzionamento che sta alla base di questi template engine è utile fare degli esempi.

Questo esempio mostra il rendering di due variabili.

```
// oggetto JSON contenente i dati
var dati = { "name": "Jon", "age": 35};
// template HTML con l'aggiunta del tag mustache
var template = "<h1>Il mio nome è {{name}} e ho {{age}} anni.</h1>";

// il risultato del rendering sarà:

Il mio nome è Jon e ho 35 anni.
```

Codice 2.1: Esempio di template rappresentante una variabile.

In questo esempio viene definito il template per rappresentare una lista di prodotti.

```
// oggetto JSON contenente i dati
var dati = prodotti: {
            "prodotti": [
                  { "name": "pane" },
                  { "name": "pasta" },
                  { "name": "biscotti" }
              ]
          };
// template HTML con l'aggiunta del tag mustache
var template = "<p>Lista della spesa:
          <ul>
            {{#prodotti}}
              <li>{{name}}</li>
            {{/prodotti}}
          </ul>
        </p>";

// il risultato del rendering sarà:

Lista della spesa:
- pane
- pasta
- biscotti
```

Codice 2.2: Esempio di template rappresentante una sezione.

2.2 Mustache.js

Mustache può essere considerato come il papà dei template system, è open-souce e logic-less e presenta implementazioni per i più famosi linguaggi di programmaione, come Java, Phyton, Ruby, PHP, JavaScript e molti altri.
Mustache.js è un implementazione per JavaScript del template system Mustache.
La libreria è molto leggera e versatile visto che permette il rendering sia lato server che lato client.
Le funzioni offerte sono *render* e *parse*, la prima si occupano di creare la stringa HTML contenente il template renderizzato partendo dai dati JSON e dal template HTML e la seconda è opzionale e permette di preparare il template in modo da velocizzare l'operazione di render.
Mustache.js viene definita logic-less perché non presenta nessun tipo di costrutto *if-then-else* e loop come *for* o *do-while*

2.2.1 Come funziona

Il suo funzionamento è molto semplice.
Per prima cosa bisogna includere la libreria all'interno della pagina HTML in cui si vuole inserire il template.
In seguito all'inclusione basta richiamare la funzione *render* passandogli l'oggetto JSON contenente i dati e il template HTML che dovrà visualizzarli.
Il template passato alla funzione sarà formato da codice HTML arricchito dai **tag** mustache necessari.
La funzione restituisce una stringa rappresentante il template renderizzato che dovrà essere inserito nella pagina.
Sfortunatamente Mustache.js non offre strumenti per la manipolazione del DOM per cui l'inserimento dovrà essere fatto dal programmatore utilizzando funzioni offerte dallo standard JavaScript o da altre librerie come JQuery.

2.2.2 Pregi e difetti

Uno dei pregi principali è sicuramente la semplicità e la leggerezza della libreria. Inoltre il set di tag offerti permette di creare template di una certa complessità e l'operazione di rendering è immediata. Come contro si può citare l'impossibilità di creare funzioni aggiuntive (helpers) per la gestione dei template, possibilità offerta da altre librerie e la totale mancanza di strumenti per la manipolazione del DOM e dei dati del template che costringe il programmatore ad appoggiarsi ad altre librerie. I template una volta renderizzati sono statici e un cambiamento nei dati non ha effetto sulla loro rappresentazione.

2.2.3 Prototipo

2.3 HandlebarsJS

HandlebarsJS è una libreria costruita sopra a Mustache quindi offre tutte le funzionalità di quest'ultimo e aggiunge al normale set di tag anche costrutti di controllo come l'espressione *if* e iteratori come *each*. La libreria offre anche un set di metodi globali che permettono al programmatore di effettuare varie operazioni sul template e i suoi dati, più la possibilità di creare delle funzioni personalizzate che possono essere inserite in uno spazio globale e riutilizzate a piacere su diversi template. Handlebare permette di precompilare il template e offre prestazioni migliori rispetto a Mustache.

2.3.1 Come funziona

Il funzionamento di HandlebarsJS è simile a quello di Mustache, avendo a disposizione l'oggetto JSON contenente i dati e il template, prima si compila il template tramite il metodo *Handlebars.compile(template)*, il risultato della compilazione è una funzione che richiamata passandole come parametro l'oggetto JSON provvede ad interpolare i dati con il template e restituisce una stringa HTML che dovrà essere inserita nella pagina. Anche in questo caso l'inserimento del template nella pagina deve essere fatto utilizzando strumenti esterni alla libreria perché essa non offre funzioni adeguate.

2.3.2 Pregi e difetti

Anche per Handlebars la leggerezza della libreria e la velocità nel rendering è da considerare un pregio. Inoltre la possibilità di sfruttare un set di metodi e di poterne creare di propri risulta un vantaggio rispetto a Mustache. Come Mustache i template una volta creati sono statici e una modifica dei dati non causa un aggiornamento del template che deve essere nuovamente ricompilato

2.3.3 Prototipo

2.4 Ractive.js

2.4.1 Come funziona

2.4.2 Pregi e difetti

2.4.3 Prototipo

2.5 Confronto finale

2.6 Libreria scelta

Capitolo 3

Strumenti e tecnologie legate alla libreria Ractive.js

Capitolo 4

I template

4.1 Ractive.js

4.1.1 L'oggetto Ractive

4.1.2 Le opzioni principali

4.1.3 La sintassi mustaches

4.1.4 Il two-way data biding

4.1.5 Gli eventi

4.1.6 Il virtual DOM

4.1.7 Creazione di un template

4.1.8 Plug-in di terze parti

4.2 Struttura dei template

4.3 Rendere il template responsive

4.4 Inserimento plug-in jQuery nei template

4.5 Caricamento dei template nelle pagine HTML

Capitolo 5

Analisi dei Requisiti

5.1 Applicazione per la modifica dei template

5.1.1 Visualizzazione lista dei template

5.1.2 Visualizzazione template selezionato

5.1.3 Editor per la modifica del template

5.2 Requisiti individuati

5.3 Riepilogo requisiti

Capitolo 6

Progettazione

6.1 Suddivisione template

6.2 Caricamento template

6.3 Creazione lista template

6.4 Visualizzazione template selezionato

6.5 Editor per la modifica del template

Capitolo 7

Realizzazione

Capitolo 8

Conclusioni

In questo capitolo finale vengono tratte le conclusioni riguardo alle attività svolte durante il periodo di stage.

8.1 Valutazione del risultato e di Ractive.js

8.1.1 Requisiti soddisfatti

8.2 Criticità

8.3 Conoscenze acquisite