



Università degli Studi dell'Aquila
Facoltà di Ingegneria

Tesi di Laurea Specialistica in Ingegneria Informatica e Automatica

Realizzazione di un prototipo della versione Cloud SaaS della suite IBM BigFix: Automazione del deployment e del testing

Relatore interno:

Prof. Serafino Cicerone

Laureando:

Beniamino Negrini

Correlatore:

Prof. Dott. Marco Secchi

Relatore esterno:

Dott. Bernardo Pastorelli

Anno Accademico 2016/2017

Dedica a piè pagina

Indice

1 CapitoloEsempio	5
2 Introduzione	6
3 IBM BigFix	7
3.1 BigFix	7
3.1.1 Architettura	7
3.1.2 BigFix Platform	10
3.1.3 BigFix Applications	10
3.1.4 Fixlets	10
3.2 IBM	11
3.3 SaaS Exploration Project	12
3.3.1 Il framework SCRUM	12
3.3.2 Sistemi di controllo di versione	13
3.3.3 RTC	14
4 CapitoloEsempio	15
A Tecnologie Utilizzate	16
A.1 Linguaggi di programmazione	16
A.2 Linguaggi di Markup e Stile	16
A.3 Framework	16
A.4 Ambiente di Sviluppo	17

A.4.1	Eclipse	17
A.4.2	Piattaforma Web	17
A.4.3	Browser Testing	17

Capitolo

1

Capitolo

Esempio

Capitolo introduttivo di prova

- item di prova 1
- item di prova 2

Paragrafo Paragrafo di prova

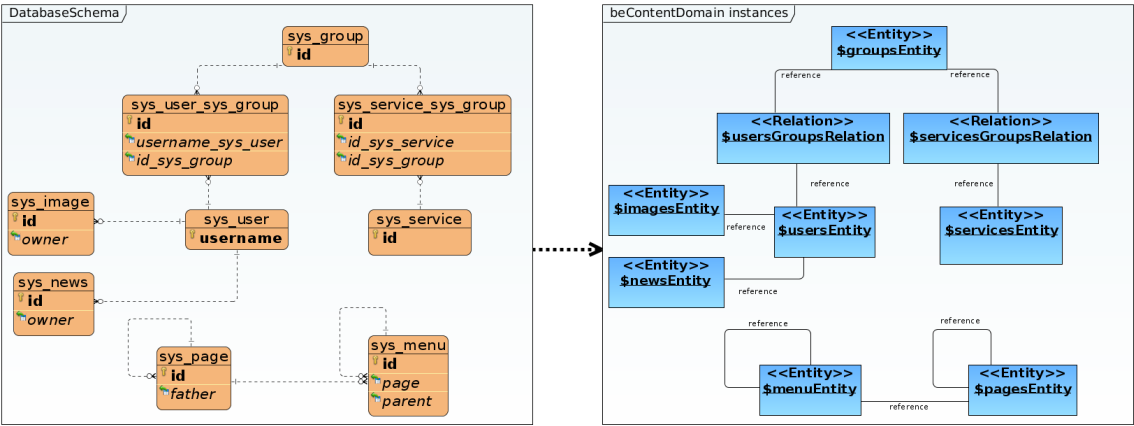


Figura 1.1: Immagine di prova

Capitolo 2

Introduzione

E' sempre più evidente che il cloud computing è il futuro del software. La rivoluzione consta nella distribuzione dei servizi di calcolo e alla virtualizzazione delle risorse. Viene data così all'utente la sensazione di un utilizzo centralizzato. Tutto ciò si è reso realizzabile dal momento in cui l'accesso alla rete è divenuto possibile da sempre più dispositivi con velocità di connessione sempre maggiore.

la tematica del cloud computing è stata centrale nel mio lavoro di tesi presso l'azienda IBM (International Business Machines Corporation), nella sua sede di Roma. Ho partecipato attivamente alla realizzazione di un prototipo software, ossia la versione SaaS della suite IBM BigFix. BigFix è una suite di prodotti dedicati alle aziende che risolvono problematiche di Endpoint Security e di compliance di dispositivi interni a determinate politiche aziendali. Tramite questi prodotti si ottiene pieno controllo su tutti i dispositivi aziendali, si rilevano eventuali attacchi, si distribuiscono aggiornamenti e patch, fino anche alla gestione energetica.

La sfida da me raccolta è quindi proprio quella di portare tutto questo arsenale di strumenti nella leggerezza del cloud. Rendendolo disponibile, nel giro di pochi minuti, anche a chi è sempre stato intimorito dalla difficoltà di installazione di uno strumento così potente, ma allo stesso tempo complesso.

Capitolo 3

IBM BigFix

3.1 BigFix

I prodotti della suite IBM BigFix consentono di monitorare e gestire in tempo reale un elevato numero di dispositivi fisici e virtuali connessi (fino a 250.000). Questi possono essere sia fisici che virtuali, come ad esempio server, desktop, notebook, dispositivi mobili, tablet, POS, ATM, chioschi self-service. Gli utenti principali di questi prodotti sono gli amministratori di sistema. Tramite le applicazioni BigFix possono avere il pieno controllo sugli endpoint, come distribuire software, applicare delle patch, effettuare il deploy di sistemi operativi, proteggere da attacchi di rete e molto altro.

3.1.1 Architettura

Un'architettura di BigFix è, per sua natura, molto articolata, poichè la necessità è quella di gestire un numero elevato ed eterogeneo di dispositivi. Essa si basa sul consolidato pattern stilistico Client/Server, ma con una struttura leggermente variata, prevedendo l'inserimento di un ulteriore layer frapposto tra client e server, i relay, i quali sono fondamentali per bilanciare il carico.

Ma partiamo subito con un esempio per avere un ponto di riferimento. Come possiamo notare, l'elemento fondamentale è il server, il quale ha lo scopo di racco-

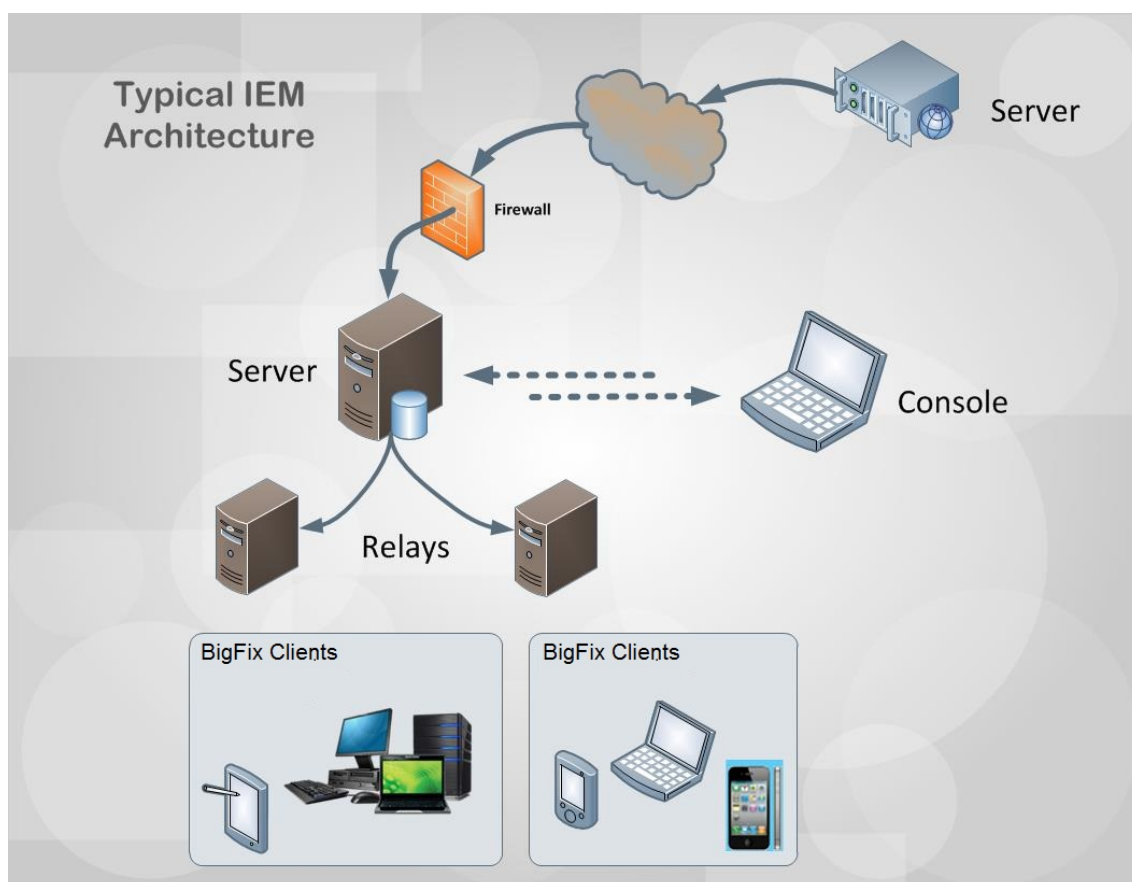


Figura 3.1: Un'architettura BigFix di esempio

gliere dei particolari messaggi, le Fixlet. Questi messaggi possono essere visualizzati dall'operatore che lavora sulla console e inoltrati quindi, a questo punto ai relay. E' competenza dei relay poi interagire con i singoli client e assicurarsi l'esecuzione delle Fixlet. Le Fixlet, infatti, altro non sono che delle azioni che devono essere necessariamente compiute dai client. Andiamo ad analizzare le singole componenti dell'architettura.

Servers Il server coordina tutto il flusso di informazioni e si preoccupa di salvare le informazioni sul database. Al tempo stesso però, lascia agli Agent il compito di effettuare analisi ed eseguire azioni specifiche. Ciò consente di liberare il Server da un pesantissima computazione. Per questo motivo il Server stesso può gestire al altissimo numero di client.

Relays I Relay si comportano come una cache tra i Client e il Server e sono di numero variabile in base al numero di Client. Aiutano il Server a gestire i dispositivi anche se funzionalmente non sono altro che Client che sono stati promossi a Relay, aggiungendo a loro dei servizi. A questo punto i Client non si interfaceranno mai con il Server, alleggerendone così notevolmente il workload. Possono, ad esempio, più Client richiedere un download al Relay, il quale effettuerà un'unica richiesta al Server

Agents Un Agent è installato su ogni Client facente parte dell'architettura di BigFix. essi hanno il compito di raccogliere le Fixlet, tramite le quali sono in grado, ad esempio, di individuare e correggere le exposure di sicurezza, configurazioni scorrette e altre vulnerabilità. Fa dei continui check per confrontare lo stato del dispositivo con le policy stabilite. Appena scopre che il dispositivo è fuori dalla compliance, viene informato il Server ed agisce subito per porre rimedio e, al termine, informa nuovamente sull'esito dell'operazione.

Web Reports I Web Reports sono il componente che consentono ad utenti autorizzati di monitorare tutti i dispositivi di BigFix. Si può in questo modo tenere traccia di vulnerabilità, azioni richieste e molto altro.

Consoles La Console permette agli amministratori di interagire con tutti i Client dell'ambiente BigFix. Gli utenti possono così distribuire velocemente patch e configurazioni.

BigFix, da un punto di vista logico, si suddivide in due grandi macro-componenti, la Platform e le Applications. La prima svolge la funzione di layer sulla quale vengono sviluppate tutte le funzionalità dello strato di applications. Questa suddivisione consente una chiara suddivisione delle competenze da parte di progettisti, sviluppatori, tester e assistenti dei clienti. Il team della platform si concentra quindi nel fornire una solida infrastruttura al team delle applications, il quale svilupperà i singoli strumenti al servizio dell'utente.

3.1.2 BigFix Platform

La Platform è una tecnologia multi-layer scritta in linguaggio C++ che agisce come colonna portante di tutta l'infrastruttura di BigFix. La Platform svolge infatti funzioni fondamentali, spesso utilizzate anche da altre applicazioni dei layer superiori. La Platform divide le responsabilità.

Lifecycle Management Include il controllo remoto, la distribuzione dei software e il deploy di sistemi operativi.

Patch Management Consente l'applicazione e la gestione delle patch.

Core Protection E' il cuore di tutta la routine di Security. Troviamo qui delle funzionalità anti-malware, firewall e protezione da qualsiasi variante di virus.

Inventory Raccoglie informazioni peculiari a riguardo dei software installati su ogni dispositivo. E' in grado di effettuare analisi sugli utilizzi fornendo una base per la decisione sulle licenze da acquistare.

Server Automation Fornisce un Hypervisor per le Virtual Machines. Esso controlla infatti eventuali malfunzionamenti degli ambienti virtuali.

3.1.3 BigFix Applications

Tutti i prodotti applicativi che fanno parte di questo componente consentono di gestire in maniera semplice tutte le operazioni inerenti alla security. A differenza della Platform sono implementate in linguaggio Java ed hanno funzionalità atomiche tra di loro. Sono l'interfaccia principale con il quale interagisce l'amministratore aziendale.

3.1.4 Fixlets

Le Fixlet sono il metodo attraverso il quale si svolgono tutte le operazioni come distribuzione di software, installazioni di patch e configurazioni. Esse sono dei mes-

saggi inoltrati ai client di BigFix e utilizzano un linguaggio di query specifico, il Relevance.

Il linguaggio Relevance

Con una Fixlet si può anche ispezionare un desiderato aspetto di un client. Il linguaggio di Relevance consente di interrogare il client identificandone caratteristiche dell'hardware o del software tramite gli Inspectors. Una necessità può essere infatti quella di applicare una Fixlet solamente a dei client con determinate caratteristiche hardware/software oppure che si trovano in stati ben definiti. Si può in questo modo, facilmente identificare il corretto sottoinsieme di client ai quali è destinata una nuova Fixlet ed applicarla solo ad essi.

3.2 IBM

Il lavoro di tesi si è svolto nell'ambito di un progetto formativo stipulato tra l'Università dell'Aquila e IBM Italia Spa. Questo progetto ha previsto un tirocinio svolto nella sede di Roma con obiettivo: "Esplorazione e prototipazione di metodi per portare prodotti BigFix su cloud", per l'appunto la realizzazione del prototipo di BigFix SaaS.

La storia dell'IBM ha inizio nei primi decenni del novecento, ma è dagli anni settanta che entra nel mercato dell'informatica, soprattutto nel settore hardware. Negli ultimi venti anni il business si è spostato sempre più sul software. In particolare soluzioni cognitive e piattaforme cloud.

IBM Security Presso la sede di Roma presente il più importante laboratorio IBM italiano, il Rome Software Lab. Nella divisione italiana ci si concentra prevalentemente sullo sviluppo back-end. Una grossa fetta del laboratorio si occupa della divisione Security di IBM. Il portfolio di Security contiene prodotti che si occupano di diversi aspetti della security aziendale, tra questi BigFix è uno dei più consolidati.

3.3 SaaS Exploration Project

Lo scopo del progetto al quale ho partecipato con il mio lavoro è quello di esplorare le tecnologie esistenti e realizzare il prototipo della versione SaaS di BigFix. A questo scopo, oltre a me, sono state allocate altre tre persone full time al conseguimento del progetto, sotto la guida dell'architect Bernardo Pastorelli.

3.3.1 Il framework SCRUM

Il team adotta il framework agile SCRUM. Questo modo di operare è di sempre maggiore diffusione ed è basato su un approccio iterativo e incrementale nello sviluppo software. Il design e lo sviluppo sono divise in iterazioni, denominate "sprint", della durata fissa di due settimane. Queste due settimane terminano sempre con una versione funzionante del prodotto, il quale viene mostrato in una demo che ne evidenzia le nuove features implementate.

SCRUM utilizza un approccio empirico alla progettazione. La filosofia di fondo del framework quella che la conoscenza deriva dall'esperienza, e quindi tutte le scelte che si prendono nel corso della progettazione devono avvenire alla luce di una sempre maggiore esperienza, la quale si ottiene avendo a disposizione il prima possibile un sottoinsieme del prodotto testabile ed usabile a se stante. Dì quì l'approccio fortemente iterativo e incrementale massimizzando le opportunità di feedback.

All'inizio del progetto vengono definiti i requisiti del prodotto (item), i quali vengono da un'attenta analisi dei bisogni dell'utente. Ogni bisogno viene modellato con una Epica, che a sua volta viene prioritizzata e aggiunta al product backlog che le indicizza. Le Epiche vengono poi scomposte in User Stories, le quali si suddividono a loro volta in Task, ossia l'elemento atomico del progetto la cui implementazione viene presa in carico da un singolo componente del team.

L'inizio di uno sprint è sempre caratterizzato da un meeting in cui si pianificano gli obiettivi per il prossimo sprint. In questo contesto si fa sempre riferimento al backlog incentrandosi sulle user stories ancora non coperte. Si cerca quindi di suddividersi i task in modo tale da avere a fine sprint quelle nuove funzionalità

usabili e dimostrabili. Demo che viene svolta sempre con la presenza di tutto il team e anche dei colleghi americani.

3.3.2 Sistemi di controllo di versione

Da un'organizzazione di questo tipo ne scaturisce la necessità di tool di controllo di versione che permettano una fluida gestione del codice e della programmazione in parallelo tra i diversi componenti del team.

GitLab A tal scopo si è adottato ormai da tempo, da tutto il team di BigFix, il software di controllo di gestione distribuito Git e una repository aziendale che consiste in una versione enterprise ad hoc per IBM di GitLab, un hosting service molto simile a GitHub.

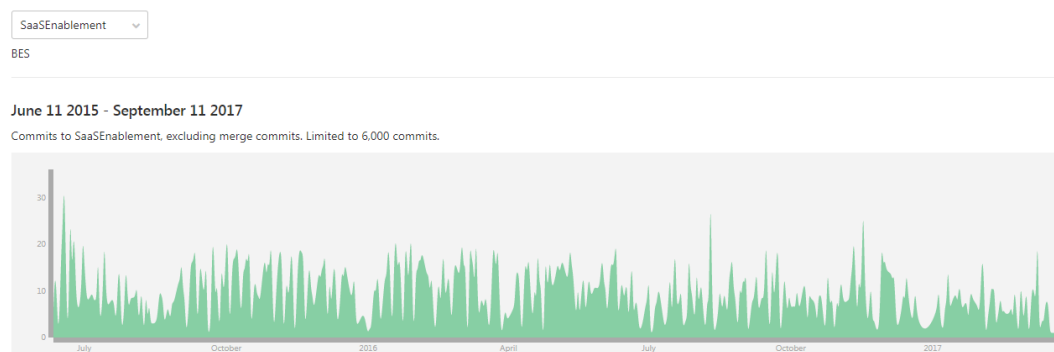


Figura 3.2: Panoramica degli ultimi contributi nel branch del progetto SaaS

Il flusso di lavoro è il seguente. Quando inizia il proprio task, il componente del team si pone su un proprio branch personale sul quale effettua i propri commit. Al termine del task viene fatta una merge request sul branch principale, sono una volta che si è testato il codice, per aggiungere i propri contributi al progetto. A questo punto, dopo una review effettuata da componenti del team accreditati, il nuovo branch verrà mergiato con il branch principale.

3.3.3 RTC

E' ovviamente necessario un tool che coordini anche la suddivisione dei task all'interno del team. A tal fine abbiamo utilizzato un software interno di IBM, ovvero Rational Team Concert (RTC). Esso offre comodi strumenti di agile planning e gestione di ciclo di vita del software. Ogni componente può così tracciare facilmente le aree di sua competenza. E' possibile inoltre usufruire di tool per il source control, controllo dei difetti e gestione delle build.

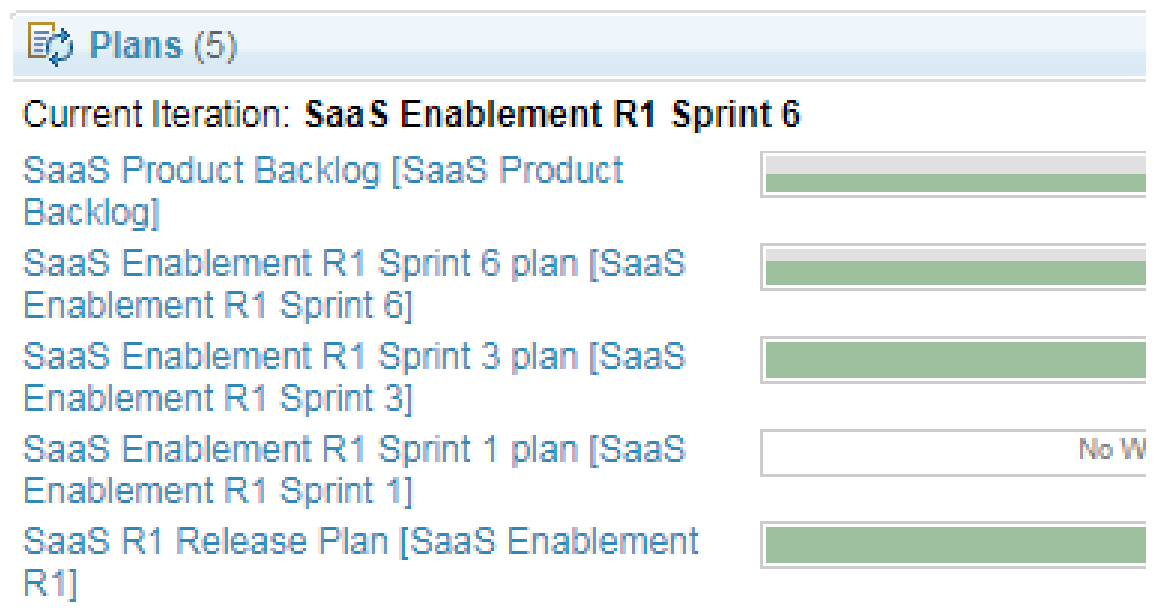


Figura 3.3: RTC. Un'esempio di come viene monitorato il completamento dei diversi sprint

CapitoloEsempio

Capitolo introduttivo di prova

- item di prova 1
- item di prova 2

Paragrafo Paragrafo di prova

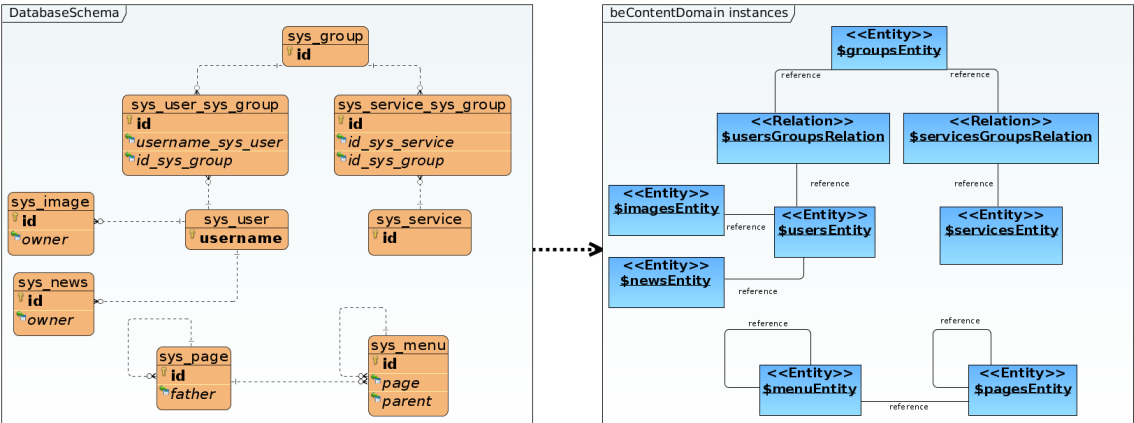


Figura 4.1: Immagine di prova

Tecnologie Utilizzate

A.1 Linguaggi di programmazione

- PHP 5.4.7
<http://www.php.net/>;
- Javascript
<http://www.w3.org/standards/webdesign/script>;

A.2 Linguaggi di Markup e Stile

- HTML4/HTML5;
- CSS/CSS3;

A.3 Framework

- Smarty Template Engine
<http://www.smarty.net/>;
- JQuery
<http://jquery.com/>;

- JQueryUI
<http://jqueryui.com/>;
- beContent
<http://www.becontent.org/>;

A.4 Ambiente di Sviluppo

A.4.1 Eclipse

Per Eclipse sono state utilizzate due versioni differenti, la 4.2.2 in ambiente Windows e la 3.8.0 in ambiente Ubuntu/Linux

<http://www.eclipse.org/>

Inoltre è stato utilizzato il pacchetto

- PHP Development Tools 3.1.1
<http://projects.eclipse.org/projects/tools.pdt>;

A.4.2 Piattaforma Web

XAMPP

<http://www.apachefriends.org>

- Apache Web Server ver. 2.4.3
<http://httpd.apache.org/>;
- MySql Database Management System ver. 5.5.27
<http://dev.mysql.com/>;

A.4.3 Browser Testing

Mozilla Firefox

- Firebug ver 1.11.2
<http://getfirebug.com/>

- Plug-In Validator ver. 0.0.6
<https://addons.mozilla.org/it/firefox/addon/validator/>;
- Plug-In Google Page Speed ver. 2.0.2.3
<https://developers.google.com/speed/pagespeed/?hl=it-IT>;

Google Chrome

- Strumenti per gli sviluppatori integrati

Responsive Testing

- Viewport Resizer- Responsive Design Bookmarklet
<http://lab.maltewassermann.com/viewport-resizer/> ;

Dedica a fine pagina