CAULDRON

Dominici Lorenzo – Vatamanelu Silviu Leonard

Giugno - Settembre 2023

Contenuti

[Modifiche Interfaccia Node-RED di Snap4City 2](#_Toc144722366)

[Porting alle versioni 2.2.3 e 3.0.2 3](#_Toc144722367)

[Integrazione Progetto CAULDRON 4](#_Toc144722368)

[Dockerfiles 7](#_Toc144722369)

[Bug Riscontrati e risolti 7](#_Toc144722370)

Introduzione

**CAULDRON**: Capacitating Agile Users with Live Debugging Resources On Node-RED, è una versione custom di node-red che offer l’utilizzo di una suite di feature per il debug e il monitoraggio di nodi e eventi di un flusso. Il nostro lavoro è consistito nel comprendere il funzionamento di tale suite, adattarla alle versioni 2.2.3 e 3.0.2 di Node-Red e di integrarla con le funzionalità di Snap4City in un unico pacchetto, disponibile sia sotto forma di immagini Docker eseguibili che di codice sorgente.

Modifiche Interfaccia Node-RED di Snap4City

Le modifiche all’interfaccia di Node-RED consentono di importare nelle iotapp i nodi creati all’interno del Resource Manager (normali e di Data Analytics). Consentono anche di importare ed esportare nel Resource Manager i flussi di un utente in modo da pubblicarlo e renderlo disponibile per altri utenti. Tali modifiche sono presenti per la versione 1.1.3 di Node-RED.

Immagine che contiene schermata, testo, software

Descrizione generata automaticamente

Immagine che contiene testo, schermata, numero, software

Descrizione generata automaticamente

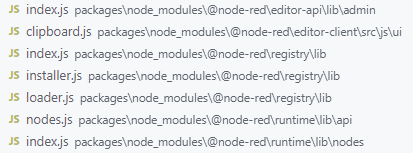
Porting alle versioni 2.2.3 e 3.0.2

Le operazioni necessarie per aggiornare Snap4City alla versione Node-RED 3.0.2 si limitano a trasferire tutte le sezioni di codice comprese tra i commenti //S4C nella versione ufficale di Node-RED 3.0.2, di fatti, questo si limita a coinvolgere esclusivamente la cartella **packages/node-modules/@node-red**. In particolare, i seguenti file modificati nella versione 1.1.3:

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Risultano nelle seguenti modifiche:



Si può subito notare che le modifiche al file **red.js** devono essere riportate all’interno del file **clipboard.js** dato il cambio di gestione del modulo **@editor-client.** I file **S4Cnodes.js** e **S4CAuth.js** non compaiono in quanto non contengono i commenti //S4C ma sono comunque presenti nelle versioni aggiornate.

Integrazione Progetto CAULDRON

Tutte le modifiche apportate dal progetto CAULDRON, anche loro limitate alla cartella **@node-red**, sono comprese tra commenti //CAULDRON. In particolare, le modifiche riguardano i seguenti file:

A screenshot of a computer

Description automatically generated

La logica principale del progetto è contenuta nei file all’interno della cartella **packages\node\_modules\@node-red\editor-client\src\js\monitoring**. Per una documentazione più completa del progetto, leggere la documentazione ufficiale al seguente link <https://repositorio-aberto.up.pt/bitstream/10216/128964/2/414763.pdf>.

In sintesi, l’integrazione del progetto CAULDRON offre la possibilità di monitorare il passaggio dei messaggi entranti nei nodi di Node-RED, fornendo anche la possibilità di utilizzare breakpoint per bloccare il flusso attraverso un nodo per poterlo analizzare, farlo procedere step-by-step, rilasciare la coda di messaggi trattenuti dal breakpoint o cancellarla.

Nello specifico, i nodi della versione CAULDRON appaiono come:

A blue rectangle with black text

Description automatically generated

Facendo click sull’icona di debug, l’aspetto del nodo cambia in:

A screen shot of a graph

Description automatically generated

Ovvero viene rivelato il grafico degli ultimi dieci messaggi che sono entrati nel nodo. Il grafico disegna linee nel caso in cui il campo tracciato continene un intero per evidenziare l’andamento del valore, altrimenti mostra punti in modo da evidenziare l’andamento temporale.

Il click sulla freccia verso il basso modifica ulteriormente l’interfaccia:

A screenshot of a computer

Description automatically generated

Vengono rivelate la textbox per il tracciamento delle variabili, che verrà spiegata in seguito, una label che indica l’ultimo messaggio e il suo tempo di arrivo.

Nel caso di un nodo inserito all’interno di un flusso, quindi con entrate e uscite, l’interfaccia rivela altre icone:

A screenshot of a computer screen

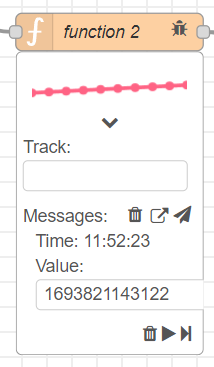
Description automatically generated

Le tre icone accanto alla label messages hanno le seguenti funzionalità:

* Quella a forma di cestino permette di cancellare tutta lo storico dei messaggi ricevuti e inviati dal nodo;
* Quella a forma di espandi permette di aprire un pop-up ch mostra lo storico dei messaggi ricevuti e inviati dal nodo;
* Quella a forma di aeroplano permette di iniettare nel nodo un messaggio personalizzato.

L’icona di pausa mette il nodo in breakpoint, consentendo di mettere in pausa il flow. I messaggi che arrivano nel frattempo vengono inseriti in una coda.

Un nodo in breakpoint ha il seguente aspetto:



Il campo Value passa a essere una textbox modificabile contenente il json del primo messaggio all’intero della coda di debug.

Le tre icone in fondo al dropdown menu permettono di gestire la coda di debug:

* Quella a forma di cestino permette di cancellare l’intero contenuto della coda,
* Quella a forma di play permette di togliere il nodo dallo stato di pausa e di svuotare gradualmente la coda dei messaggi,
* Quella a forma di Fast Forward permette di estrarre il primo messaggio dalla coda e di fargli proseguire il flusso.

Una particolare aggiunta alla l’attuale versione di Cauldron permette di selezionare, attraverso una apposita textbox presente in ogni nodo, l’attributo da tracciare. La sintasi ammessa è quella usuale del Javascript quindi per un oggetto del tipo:

{

A:[

{

B:’0b’,

},

{

B:”1b”,

}

]

}

Digitare nella textbox l’espessione A[0].B mostra il valore ‘0b’ mentre l’espressione

A[1].B mostra il valore ‘1b’. Se l’espressione non corrisponde a nessun attributo dell’oggetto non vien mostrato alcun risultato.

Tale funzionalità può essere usata in modalità statica(‘static’) o dinamica(‘dynamic’): per modalità statica si intende che il cambio della textbox ha effetto dal prossimo messaggio che attraversa il nodo, mentre per modalità dinamica si intende che l’aggiornamento della textbox comporta un aggiornamento istantaneo del valore visualizzato a schermo.

Se la textbox rimane vuota, il valore visualizzato sarà quello di default, solitamente timestamp di arrivo del messaggio per i nodi generici.

Le feature appena discusse sono accessibili tramite una UI dropdown propria di ogni nodo, attivabile tramite il rispettivo pulsante di debug presente sul nodo.

Ai fini dell’estensione del grafico, si fa presente che esso memorizza e tiene di conto solo gli ultimi 10 messaggi ricevuti, e NON lo storico completo dall’attivazione del flusso.

Dockerfiles

Nella sotto-cartella **dockerfiles** sono presenti 6 Dockerfile per comporre le immagini di riferimento ovvero l’aggiornamento alla versione 3.0.2 di Node-RED con le modifiche di snap4city, l’aggiornamento alla versione 3.0.2 di Node-RED con snap4city e cauldron e l’ aggiornamento alla versione 2.2.3 di Node-RED con snap4city e cauldron. Per ogni scenario sono presenti due versioni: basic o user e advanced o developer.

I dockerfile per le versioni advanced utilizzano come base l’immagine **disitlab/snap4city-nodered-v2.2.2-adv:v10** mentre le versioni basic utilizzano l’immagine **disitlab/snap4city-nodered-v2.2.2-basic:v7**.

Tutti i dockerfile hanno la stessa struttura: viene eseguito un clone del tag riferito alla versione desiderata dalla repository **lorenzo-dominici/cauldron** su github dove è hostato il progetto. Vengono installate le dipendenze di cauldron e viene effettuato il build del progetto. In seguito viene copiata la directory **packages/node\_modules/@node-red** all’interno della cartella che contiene l’eseguibile (nella versione 3 è necessario copiare anche il modulo node-watch) e viene lanciato l’eseguibile con l’istruzione **npm start –userDir /data**.

Le immagini, inoltre, sono presenti su docker hub per un utilizzo immediato con i seguenti nomi:

- ipsdisit/snap4city-cauldron-v3.0.2-adv:v1

- ipsdisit /snap4city-cauldron-v3.0.2-usr:v1

- ipsdisit /snap4city-v3.0.2-adv:v1

- ipsdisit /snap4city-v3.0.2-usr:v1

- ipsdisit /snap4city-cauldron-v2.2.3-adv:v1

- ipsdisit /snap4city-cauldron-v2.2.3-usr:v1

# Bug Riscontrati e risolti

Rispetto alla repository originale, locata al seguente link: <https://github.com/diogotorres97/node-red>, le problematiche riscontrate e successivamente risolte, locate al seguente link: <https://github.com/lorenzo-dominici/cauldron> e poi copiate al link: <https://github.com/ipsdisit/cauldron>, consistono in:

* Rendere funzionante il dockerfile originale per lo sviluppo dell’applicazione;
* Importazione componenti e modifiche di snap4city;
* Risoluzione delle problematiche di integrazione del sistema di breakpoint, in quanto in conflitto con variabili pre-esistenti in node-red, in particolare il conflitto riguardava una collisione di nomi tra la funzione debug della classe Node e della proprietà debug aggiunta da CAULDRON. Questo ha risolto anche le problematiche relative alla funzionalità repeat dei nodi inject;
* Aggiunta del tracciamento di singole variabili all’interno dei messaggi, mediante apposito campo da compilare che viene interpretato all’arrivo di un messaggio;
* Risoluzione delle problematiche relative alla coerenza del comportamento dell’applicazione nel caso di cambio di contesto, ovvero navigazione tra i flow del workspace, in quanto il progetto originale, al cambio di workspace, non iniettava le icone di debug nei nodi. Risolto aggiungendo un iniezione dell’HTML al ricevimento del’evento relativo al cambio di workspace;
* Risoluzione delle problematiche relative alla coerenza del comportamento dell’applicazione nel caso di chiusura della pagina, in quanto il progetto originale, alla chiusura dell’interfaccia di node-red bloccava l’esecuzione di tutti i nodi. Risolto applicando una logica coerente con il comportamento richiesto;
* Risoluzione dei problemi di compatibilità di Snap4city dovuti all’aggionamento di node-red alla versione 3, in particolare tutte le funzioni che usavano la libreria “when” ora usano la classe Promise e, inoltre, è stato copiato un metodo dalla versione 2 di node-red, rimosso nella versione 3, per garantire la compatibilità dei metodi;
* Risoluzione di problemi dovuti alla visualizzazione dei grafici di debug contenuti nei singoli nodi, risolti applicando il filtro di tracciamento ai valori da disegnare sul grafico;
* Modifica del comportamento dell’applicazione alla chiusura, ovvero la disabilitazione di tutti i breakpoint e il consumo graduale delle code accumulate;
* Consistenza delle icone e del comportamento delle feature di debug nel caso di deploy di un flusso pre-esistente, in quanto l’operazione di deploy andava a cambiare automaticamente lo stato di debug, senza aggiornare le icone.