NORRIS Framework

FLAMETECH Inc.

Manuale Sviluppatore

Informazioni sul documento

Versione	2.0.0
Redazione	Merlo Gianluca
Verifica	Sartor Michele
Responsabile	Zanetti Davide
Uso	Esterno
Lista di distribuzione	**FlameTech Inc.**
	Prof. Vardanega Tullio
	Prof. Cardin Riccardo
	CoffeeStrap

Descrizione

Manuale sviluppatore per l'utilizzo del framework Norris

Versione	Modifica	Autore	Ruolo	Data	Stato
2.0.0	Approvazione documento	Zanetti Davide	Responsabile	2015/06/11	Approvato
1.1.0	Verifica documento	Sartor Michele	Verificatore	2015/06/11	Verificato
1.0.3	Aggiornamento appendice B. segnalazioni	Merlo Gianluca	Amministratore	2015/06/08	In Lavorazione
1.0.2	Stesura sezione 4.3. esempio di progetto	Merlo Gianluca	Amministratore	2015/06/08	In Lavorazione
1.0.1	Aggiornamento sezione 3. librerie esterne	Merlo Gianluca	Amministratore	2015/06/08	In Lavorazione
1.0.0	Approvazione documento	Meneguzzo Francesco	Responsabile	2015/05/26	Approvato
0.1.0	Verifica documento	Merlo Gianluca	Verificatore	2015/05/26	Verificato
0.0.7	Terminata stesura Glossario interno	Cardin Andrea	Programmatore	2015/05/25	In Lavorazione
0.0.6	Stesura sezione Software Development Kit	Faggin Andrea	Programmatore	2015/05/23	In Lavorazione
0.0.5	Stesura sezione Installazione	Sartor Michele	Amministratore	2015/05/23	In Lavorazione
0.0.4	Iniziata stesura Glossario interno	Cardin Andrea	Programmatore	2015/05/23	In Lavorazione
0.0.3	Stesura sezione Configurazione Progetto	Cardin Andrea	Programmatore	2015/05/23	In Lavorazione
0.0.2	Stesura sezione Introduzione e Requisiti di Sistema	Sartor Michele	Amministratore	2015/05/22	In Lavorazione
0.0.1	Stesura scheletro del documento	Sartor Michele	Amministratore	2015/05/22	In Lavorazione

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento rappresenta il manuale sviluppatore del *framework*$_G$ Norris. All'interno sono esposte tutte le funzionalità fornite dal prodotto. Il manuale è suddiviso in sezioni riguardanti: l'installazione della libreria, la configurazione di un'istanza del prodotto e le funzioni utilizzabili dallo sviluppatore.

1.2 Scopo del prodotto

Lo scopo del prodotto è la realizzazione di un *framework*$_G$ per *Node.js*$_G$, compatibile con l'utilizzo standard dei *middleware*$_G$ di *Express*$_G$ in versione 4.x, per la realizzazione rapida di grafici aggiornabili in tempo reale.

2 Requisiti di sistema

Il *framework*$_G$ Norris è compatibile con tutti i sistemi operativi supportati da *Node.js*$_G$ e dal Node package manager *npm*$_G$. È necessaria l'installazione di *Node.js*$_G$ versione 0.12.2 o superiore e del Node package manager *npm*$_G$ versione 2.7.4 o superiore per l'utilizzo del *framework*$_G$.
I browser supportati sono Chrome dalla versione 38.0.X o superiore e Firefox dalla versione 32.x o superiore. Ulteriori dipendenze da librerie verranno risolte automaticamente al momento dell'installazione.
Per l'installazione è richiesto l'utilizzo di un terminale da cui sia possibile eseguire il Node package manager *npm*$_G$ ed è richiesta una connessione ad Internet.

3 Installazione

3.1 Installazione del *framework*$_G$

È necessaria l'installazione di *Node.js*$_G$ e del Node package manager *npm*$_G$ per l'utilizzo del *framework*$_G$. Per l'installazione è richiesto l'utilizzo di un terminale da cui sia possibile eseguire il Node package manager *npm*$_G$ ed è richiesta una connessione ad Internet.
Tramite il seguente comando sarà possibile installare il *framework*$_G$ Norris:

```
npm install norris-rtbi
```

È inoltre necessario installare *Express*$_G$ e *Socket.io*$_G$ tramite i comandi:

```
npm install express
npm install socket.io
```

e che sia presente nel sistema Bower per la gestione delle librerie necessarie al funzionamento della parte front-end. Nel caso non sia presente potrà essere installato tramite il seguente comando

```
npm install -g bower
```

per poter avviare un progetto. Altre dipendenze da librerie verranno risolte automaticamente al momento dell'installazione.
Per l'utilizzo di Norris si rimanda alla sezione 4.

4 Configurazione progetto

4.1 Librerie richieste e istanziazione

Per poter usufruire delle funzionalità del *framework*$_G$ Norris è richiesto l'utilizzo di due librerie che sono necessarie per il corretto funzionamento del *framework*$_G$.
È necessario disporre di un'istanza di:

- ***Socket.io***$_G$: in modo da poterne fornire un suo *namespace*$_G$ esclusivo al *framework*$_G$ Norris che verrà utilizzato per realizzare le notifiche *push*$_G$ sui singoli grafici;
- ***Express***$_G$: per poter fornire l'equivalente di un punto di *mount*$_G$ al *framework*$_G$ Norris per una corretta integrazione di quest'ultimo con l'applicazione in cui se ne vuole fare uso, e per rendere possibile la richiesta delle risorse dal web.

Come riferimento, viene presentato una corretta modalità di configurazione degli aspetti appena trattati.

```
// Express instantiation
var app = require('express')();

// Socket.io instantiation on the express instance
var http = require('http').createServer(app);
var socket = require('socket.io').listen(http);

// Creating namespace for the Norris instance
var nsp = socket.of('/norris');

// Norris instantiation
var Norris = require('norris-rtbi')(nsp);
```

4.2 Mount

Per entrare nel dettaglio dell'integrazione del *framework*$_G$ Norris con l'applicazione in cui se ne vuole fare uso, viene presentato un esempio di codice per il corretto utilizzo di un punto di *mount*$_G$ e successivamente ne verranno indicati gli effetti. Le funzionalità dell'*SDK*$_G$ del *framework*$_G$ presentate in questo esempio verranno discusse in profondità nella sezione 5.

```
// Creating Norris objects
var page = new Norris.Page("Page title", { "pageWidth": 1024, "columns": 2});
var barChart = new Norris.BarChart("Bar Chart title", "X Axis name", "Y Axis
    name", [1], [[5]]);
page.addGraph(barChart);

// Mounting the Norris object "page" on the express application
app.use('/norris', Norris.PageRouter(page));
```

Nel precedente esempio, l'ultima istruzione fornisce al *framework*$_G$ Norris un punto di *mount*$_G$ su cui effettuare il *routing*$_G$ interno per la gestione degli oggetti che sono resi disponibili all'esterno. Nel dettaglio, con una richiesta *HTTP*$_G$ all'URL dell'applicazione con l'aggiunta finale di, nel caso d'esempio, /norris visualizzerà il template di pagina *HTML*$_G$ fornito dal *framework*$_G$ con all'interno la rappresentazione dei grafici creati.

Viene inoltre esteso il *routing*$_G$ interno in modo che l'aggiunta di /raw alla fine dell'indirizzo URL di una risorsa Norris fornisca i dati grezzi della risorsa che è stata resa disponibile. In questo modo viene reso possibile utilizzare queste informazioni all'esterno del template fornito dal *framework*$_G$ Norris nel caso si preferisca creare un proprio template e utilizzare librerie grafiche diverse.

4.2.1 PageRouter(page)

Quando si crea una pagina per renderla disponibile tramite *Express*$_G$ è necessario invocare il metodo PageRouter() al quale dovrà esser passata l'istanza della pagina. Questo si occuperà di crearne un middleware che deve essere inserito nella funzione app.use(URL, middleware) come descritto nell'esempio in alto. **Nel caso non venga effettuata questa operazione la pagina non sarà visualizzabile.**

4.3 Esempio di progetto

Di seguito sarà riportato il codice completo per creare una pagina di Norris contenente due grafici, avviare l'applicazione e successivamente aggiornarne i dati dei grafici.

```
// Express instantiation
var app = require('express')();
// Socket.io instantiation on the express instance
var http = require('http').createServer(app);
var socket = require('socket.io').listen(http);
// Creating namespace for the Norris instance
var nsp = socket.of('/Socket');
// Norris instantiation
var Norris = require('norris-rtbi')(nsp);

// Creating Norris objects
var Page = Norris.Page;
var LineChart = Norris.LineChart;
var BarChart = Norris.BarChart;
var PageRouter = Norris.PageRouter;
// Create new page
var page = new Page("First page", {columns:2} );

//creating Line Chart with some options
var axisLabel = [1,2,3];
var lineData = [[4,6,5],[7,6,2],[2,4,5],[3,2,1]];
var lineOpt = { series:["Jan", "Feb", "Mar", "Jun"]
              , legendPosition: "bottom"
              , labelsLimit: 10
              };
var myLine = new LineChart("Line Chart title", "X Axis", "Y Axis", axisLabel,
    lineData, lineOpt);

//creating Bar Chart with some options
var barData = [[7,8,9],[-1,-2,-3]];
var barOpt = { orientation: "horizontal", valueType: "dollars"};
var myBar = new BarChart("Bar Chart title", "X Axis", "Y Axis", [1,2,3],
    barData, barOpt);
```

```
// add graphs into page
page.addGraph(myLine);
page.addGraph(myBar);

// Mounting the Norris object "page" on the express application
app.use('/myPage', PageRouter(page));

// Update function
var label = 4;
setInterval(function() {
        var randVal = Math.floor(Math.random()*10);
        // insert new random value on new label
        myLine.updateStream(label, [randVal, randVal*2, randVal*0.5,
            randVal*1.5]);
        // updating serie "3" with random value on various label
        myLine.updateInPlace(label-1, 3, Math.floor(Math.random()*10));
        // update label 2 on serie 1 with random value
        myBar.updateInPlace(2,0, Math.floor(Math.random()*10)+1);
        label++;
}, 5000);

http.listen(process.env.PORT || 3000);
```

5 Software development kit

5.1 Descrizione generale

Norris *framework*$_G$ mette a disposizione dello sviluppatore due entità principali: la pagina ed i grafici. La pagina è una struttura che permette di raggruppare vari tipi di grafici. I grafici sono di quattro tipi e devono essere inseriti in una pagina per essere visualizzati nella loro rappresentazione grafica. È possibile creare grafici di tipo: *Bar Chart*$_G$, *Line Chart*$_G$, *Map Chart*$_G$ e *Table*$_G$.

5.2 Pagina

Pagina è una struttura atta a contenere le proprietà ed i grafici di una pagina web all'interno del *server*$_G$. La pagina sarà disponibile on-line dopo che sarà stata aggiunta ad un punto di *mount*$_G$ di un *middleware*$_G$ di *Express*$_G$, vedi sez. 4.2, tramite il metodo PageRouter() come descritto nella sezione 4.2.1

5.2.1 Page(title, options)

Il metodo Page() permette di creare una pagina e di assegnare ad essa un titolo, una larghezza, ed un numero massimo di grafici per riga. Il parametro title è obbligatorio, mentre options è opzionale.

- **title**: prevede il passaggio di una stringa che rappresenterà il titolo della pagina web;
- **options**: permette il passaggio di un oggetto di tipo *JSON*$_G$ strutturato come nel seguente esempio. Questo parametro non è obbligatorio, è inoltre possibile inserire uno o più parametri tra quelli disponibili.

```
{ "pageWidth": value
, "columns": value
}
```

  - **pageWidth**: prevede il parametro di un numero intero maggiore di 800, che rappresenta la larghezza in pixel della pagina web;
  - **columns**: prevede il passaggio di un numero intero, che rappresenta il numero massimo di grafici che saranno visualizzati su una riga della pagina. Non possono essere inseriti valori superiori a 12, in tal caso il software terminerà. Nel caso siano aggiunti più grafici alla pagina rispetto al numero massimo di grafici per riga impostato i grafici saranno mostrati su più righe.

5.2.2 getPageInfo()

Questo metodo invocato su un oggetto di tipo Page restituisce un oggetto *JSON*$_G$ contenente tutti i dati ed i parametri riguardanti la pagina ed i grafici che sono stati aggiunti.

5.2.3 addGraph(graph)

Questo metodo invocato su un oggetto di tipo Page permette di inserire un grafico in una pagina, dando così la possibilità di essere visualizzato. Un grafico può essere aggiunto ad una o più pagine.

5.3 Line Chart

Un grafico a linee è un tipo di grafico che mostra le informazioni come una serie di punti di dati connessi da una linea. Nel grafico è possibile inserire più set di dati e impostare i parametri descritti in seguito.

5.3.1 LineChart(title, xAxisName, yAxisName, labels, data, options)

Il metodo LineChart() permette di creare un grafico di tipo *Line Chart*$_G$. I parametri sono tutti obbligatori ad esclusione di options.

- **title**: prevede il passaggio di una stringa che rappresenterà il titolo del grafico;
- **xAxisName**: prevede il passaggio di una stringa che rappresenterà l'etichetta corrispondente all'asse delle ascisse;
- **yAxisName**: prevede il passaggio di una stringa che rappresenterà l'etichetta corrispondente all'asse delle ordinate;
- **labels**: prevede il passaggio di un array di stringhe che rappresenterà le etichette corrispondenti all'asse delle ascisse;
- **data**: prevede il passaggio di dati iniziali sotto forma di array di array. Ogni elemento dell'array esterno corrisponde ad una serie di dati e a sua volta ogni elemento dell'array interno è un valore corrispondente ad una label. Gli array interni dovranno avere dimensione pari al numero di label inserite o il programma terminerà;

$$[[a_0, a_1, a_2, ..., a_{label.length-1}], [b_0, b_1, b_2, ..., b_{label.length-1}], [c_0, c_1, c_2, ..., c_{label.length-1}]]$$

- **options**: prevede il passaggio di oggetto di tipo *JSON*$_G$ strutturato come nel seguente esempio. È possibile inserire uno o più parametri tra quelli disponibili;

```
{ "series": []
, "grid": "value"
, "legend": "value"
, "legendPosition": "value"
, "colors": []
, "valueType": []
, "decimals": value
, "labelsLimit": value
}
```

  - **series**: permette l'inserimento di un array di stringhe che corrispondono ai nomi delle serie dei dati. Questi valori saranno inseriti nella legenda;

- **grid**: permette l'inserimento del valore true per visualizzare la griglia del grafico, oppure false per nasconderla. Di default la griglia viene visualizzata;
- **legend**: permette l'inserimento del valore true per visualizzare la legenda nel grafico, oppure false per nasconderla. Di default la legenda viene visualizzata;
- **legendPosition**: permette l'inserimento dei valori right, left, top o bottom, per spostare la legenda nel lato del grafico scelto. Di default la legenda viene visualizzata nel lato destro del grafico;
- **colors**: permette l'inserimento dei colori delle serie di dati presenti nel grafico. Dovrà essere impostato un colore per ogni serie di dati presente nel caso sia deciso di impostarli manualmente, mentre in caso contrario i colori saranno impostati di default da Norris.
  Ogni colore dovrà avere il formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali. I colori dovranno essere di tipo string inseriti in un array;
- **valueType**: prevede il passaggio di una stringa corrispondente a euro, -dollars o pounds per specificare il formato dei dati da visualizzare;
- **decimals**: prevede il passaggio di un intero che indica il numero dei decimali visualizzati;
- **labelsLimit**: prevede il passaggio di un intero che permette di impostare il numero di dati che saranno mantenuti nel $client_G$, impostando cosi un upper bound. Il valore di default è impostato a 300.

Nel caso venga passato un dato non conforme alle specifiche il software terminerà.

5.3.2 getChartInfo()

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo *Line Chart*$_G$, restituisce un oggetto $JSON_G$ contenente tutti i dati ed i parametri riguardanti il grafico.

5.3.3 updateInPlace(label, set, newValue)

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo *Line Chart*$_G$, permette di aggiornare un valore presente nel grafico. Tutti i parametri sono obbligatori.

- **label**: prevede il passaggio dell'etichetta corrispondente al dato che deve essere aggiornato;
- **set**: prevede il passaggio di un indice corrispondente alla serie che deve essere aggiornata, devono essere inseriti valori compresi tra 0 e n-1, con n corrispondente al numero di serie di dati presenti nel grafico;
- **newValue**: prevede il passaggio del nuovo valore da inserire nell'asse delle ordinate del grafico.

5.3.4 updateStream(newLabel, newValue)

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo *Line Chart*$_G$, permette di aggiornare il grafico aggiungendo un nuovo valore alle serie di dati. Tutti i parametri sono obbligatori.

- **newLabel**: prevede il passaggio di una stringa corrispondente al nuovo valore che deve essere inserito nell'asse delle ascisse;
- **newValue**: prevede il passaggio di un array dei nuovi valori da inserire nell'asse delle ordinate del grafico. L'array deve contenere un valore per ogni serie di dati presenti nel grafico.

5.4 Bar Chart

Un grafico a barre è un tipo di grafico che mostra le informazioni come una serie di barre verticali o orizzontali. Nel grafico è possibile inserire più set di dati e impostare i parametri descritti in seguito.

5.4.1 BarChart(title, xAxisName, yAxisName, labels, data, options)

Il metodo BarChart() permette di creare un grafico di tipo *Bar Chart*$_G$. I parametri sono tutti obbligatori ad esclusione di options.

- **title**: prevede il passaggio di una stringa che rappresenterà il titolo del grafico;
- **xAxisName**: prevede il passaggio di una stringa che rappresenterà l'etichetta corrispondente all'asse delle ascisse;
- **yAxisName**: prevede il passaggio di una stringa che rappresenterà l'etichetta corrispondente all'asse delle ordinate;
- **labels**: prevede il passaggio di un array di stringhe che rappresenterà le etichette corrispondenti all'asse delle ascisse;
- **data**: prevede il passaggio di dati iniziali sotto forma di array di array. Ogni elemento dell'array esterno corrisponde ad una serie di dati, a sua volta ogni elemento dell'array interno è un valore corrispondente ad una label. Gli array interni dovranno avere dimensione pari al numero di label inserite o il programma terminerà;

$$[[a_0, a_1, a_2, ..., a_{label.length-1}], [b_0, b_1, b_2, ..., b_{label.length-1}], [c_0, c_1, c_2, ..., c_{label.length-1}]]$$

- **options**: prevede il passaggio di oggetto di tipo *JSON*$_G$ strutturato come nel seguente esempio. Non è necessario inserire tutti i parametri descritti di seguito;

```
{ "series": []
, "orientation": "value"
, "grid": "value"
, "legend": "value"
, "legendPosition": "value"
, "colors": []
, "valueType": []
, "decimals": value
}
```

- **series**: permette l'inserimento di un array di stringhe per identificare i nomi delle serie di dati e inserire dei nomi personalizzati nella legenda. L'array dovrà avere dimensione pari al numero di serie inserite o il programma terminerà. Nel caso questa opzione non sia inserita i nomi saranno inseriti di default da Norris;
- **orientation**: permette l'inserimento del valore vertical per creare un grafico a barre verticali, oppure horizontal per creare un grafico a barre orizzontali. Di default il grafico viene creato con orientamento verticale;
- **grid**: permette l'inserimento del valore true per visualizzare la griglia nel grafico, oppure false per nasconderla. Di default la griglia viene visualizzata;
- **legend**: permette l'inserimento del valore true per visualizzare la legenda nel grafico, oppure false per nasconderla. Di default la legenda viene visualizzata;
- **legendPosition**: permette l'inserimento dei valori right, left, top o bottom, per spostare la legenda nel lato del grafico scelto. Di default la legenda viene visualizzata nel lato destro del grafico. Nel caso l'orientamento sia impostato come orizzontale la legenda non può essere visualizzata nel lato sinistro;
- **colors**: permette l'inserimento dei colori delle serie di dati presenti nel grafico. Dovrà essere impostato un colore per ogni serie di dati presente nel caso sia deciso di impostarli manualmente, in caso contrario i colori saranno impostati di default da Norris.
  Ogni colore dovrà avere il formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali. I colori dovranno essere di tipo string inseriti in un array;
- **valueType**: prevede il passaggio di una stringa corrispondente a: euro, dollars o pounds per specificare il formato dei dati da visualizzare;
- **decimals**: prevede il passaggio di un intero che indica il numero dei decimali visualizzati.

Nel caso venga passato un dato non conforme alle specifiche il software terminerà.

5.4.2 getChartInfo()

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo *Bar Chart*$_G$, restituisce un oggetto $JSON_G$ contenente tutti i dati e i dettagli riguardanti il grafico.

5.4.3 updateInPlace(label, set, newValue)

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo *Bar Chart*$_G$, permette di aggiornare un valore presente nel grafico. Tutti i parametri sono obbligatori.

- **label**: prevede il passaggio dell'etichetta corrispondente al dato che deve essere aggiornato;
- **set**: prevede il passaggio di un indice corrispondente alla serie che deve essere aggiornata, devono essere inseriti valori compresi tra 0 e n-1, con n corrispondente al numero di serie di dati presenti nel grafico;
- **newValue**: prevede il passaggio del nuovo valore da inserire nell'asse delle ordinate del grafico.

5.5 Map Chart

Un grafico mappa è un tipo di grafico, che mostra le informazioni all'interno di una mappa fornita da Google Maps. Nel grafico è possibile inserire set di dati per evidenziare dei percorsi, inserire dei *marker*$_G$ per indicare specifici punti ed impostare i parametri descritti in seguito.

5.5.1 MapChart(title, paths, points, centerLatitude, centerLongitude, options)

Il metodo MapChart() permette di costruire un grafico di tipo *Map Chart*$_G$, i parametri sono tutti obbligatori ad eccezione di options.

- **title**: prevede il passaggio di una stringa che rappresenterà il titolo del grafico;
- **paths**: prevede il passaggio di un array di array. Ogni array interno rappresenta un percorso nella mappa, ed è composto da un array di punti d'interesse. Nell'esempio sottostante la serie a rappresenta il primo percorso, mentre la serie b rappresenta il secondo. Il percorso sarà calcolato tramite le *API*$_G$ di Google Maps seguendo l'ordine di tutti i punti d'interesse forniti. Per ogni punto deve essere inserito un array di due elementi, che rappresenti rispettivamente le coordinate latitudine e longitudine.

  $$[[[x_{a,0}, y_{a,0}], [x_{a,1}, y_{a,1}]], [[x_{b,0}, y_{b,0}], [x_{b,1}, y_{b,1}], [x_{b,2}, y_{b,2}]]]$$

  È possibile creare un grafico privo di percorsi inserendo un array vuoto, ma non sarà possibile aggiungere percorsi in momenti successivi alla creazione del grafico;
- **points**: prevede il passaggio di un array di *JSON*$_G$ composto da tre campi: latitude, longitude ed id. Id deve avere un codice univoco nell'array dei punti inseriti.

  $$[\{latitude : x_0, longitude : y_0, id :'' point{-}1''\}, \{latitude : x_1, longitude : y_1, id :'' point{-}2''\}]$$

  È possibile creare un grafico privo di *marker*$_G$ inserendo un array vuoto, sarà possibile aggiungere dei punti successivamente tramite l'aggiornamento di tipo movie;
- **centerLatitude**: prevede il passaggio di un double con valori compre tra -90 e 90, questo parametro indica la latitudine del punto in cui è centrata la mappa;
- **centerLongitude**: prevede il passaggio di un double con valori compre tra -180 e 180, questo parametro indica la longitudine del punto in cui è centrata la mappa;
- **options**: prevede il passaggio di oggetto di tipo *JSON*$_G$ strutturato come nel seguente esempio. Non è necessario inserire tutti i parametri descritti sotto;

```
{ "zoom": 0
, "legend": "value"
, "mapLegendPosition": "value"
, "colors": [ ]
, "pathName": [ ]
, "pathMode": [ ]
}
```

- **zoom**: permette l'inserimento di un numero intero per impostare il livello iniziale di zoom della mappa. Sono ammessi valori compresi tra 0 e 19. Di default il livello viene impostato a 4 da Norris;
- **legend**: permette l'inserimento del valore true per visualizzare la legenda nel grafico, oppure false per nasconderla. Di default la legenda viene visualizzata;
- **mapLegendPosition**: permette l'inserimento dei valori top-right, top-left, bottom-right o bottom-left per spostare la legenda nel rispettivo angolo del grafico scelto. Di default la legenda viene visualizzata nell'angolo in alto a destra del grafico;
- **colors**: permette l'inserimento dei colori delle serie di dati presenti nel grafico. Dovrà essere impostato un colore per ogni serie di dati presente nel caso sia deciso di impostarli manualmente, nel caso contrario i colori saranno impostati di default da Norris. Ogni colore dovrà avere il formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali. I colori dovranno essere di tipo string inseriti in un array;
- **pathName**: permette l'inserimento di un array di stringhe per identificare i nomi dei percorsi ed inserire dei nomi personalizzati nella legenda. L'array dovrà avere dimensione pari al numero di percorsi inseriti o il programma terminerà. Nel caso questa opzione non sia inserita i nomi saranno inseriti di default da Norris;
- **pathMode**: prevede il passaggio di una stringa corrispondente a walking, driving, bicycling o transit per specificare il tipo di mezzo di trasporto con il quale si vuole effettuare il percorso. In particolare:
  - **walking**: se si vuole effettuare il percorso a piedi;
  - **driving**: se si vuole effettuare il percorso in auto;
  - **bicycling**: se si vuole effettuare il percorso in bicicletta;
  - **transit**: se si vuole effettuare il percorso con i mezzi pubblici.

  Questo permetterà di calcolare percorsi più specifici in base alle esigenze. Di default sarà impostato il valore driving.

Nel caso venga passato un dato non conforme alle specifiche il software terminerà.

5.5.2 getChartInfo()

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo *Map Chart*$_G$, restituisce un oggetto $JSON_G$ contenente tutti i dati e i dettagli riguardanti il grafico.

5.5.3 updateInPlace(point, latitude, longitude)

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo *Map Chart*$_G$, permette di aggiornare la posizione di un $marker_G$ precedentemente inserito nel grafico. Tutti i parametri sono obbligatori.

- **point**: prevede il passaggio di una stringa corrispondente all'identificativo del $marker_G$ che deve essere aggiornato;
- **latitude**: prevede il passaggio della nuova latitudine dove si intende spostare il punto. Il valore fornito deve essere compreso tra -90 e 90;

- **longitude**: prevede il passaggio della nuova longitudine dove si intende spostare il punto. Il valore fornito deve essere compreso tra -180 e 180.

5.5.4 updateMovie(newPositions)

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo *Map Chart*$_G$, permette di aggiornare contemporaneamente la posizione di più *marker*$_G$ presenti nel grafico. L'invocazione di questo metodo con il passaggio di un array di punti prevede l'aggiornamento dei punti esistenti con i punti presenti nell'array. Nel caso siano passati identificativi di punti non presenti, essi verranno aggiunti al grafico, mentre nel caso il grafico contenga punti con identificativi non presenti nell'array newPositions i punti saranno rimossi dal grafico e i punti già esistenti saranno spostati alle nuove coordinate.

- **newPosition**: prevede il passaggio di un array in cui ogni elemento corrisponde ad un *marker*$_G$, ogni cella dell'array contiene un *JSON*$_G$ composto come descritto sotto.

$$[\{\}, \{\}, \{\}, ...]$$

  - **id**: prevede il passaggio di una stringa che corrisponde al codice identificativo del *marker*$_G$ nella mappa;
  - **latitude**: prevede il passaggio della nuova latitudine dove si intende spostare il punto. Il valore fornito deve essere compreso tra -90 e 90;
  - **longitude**: prevede il passaggio della nuova longitudine dove si intende spostare il punto. Il valore fornito deve essere compreso tra -180 e 180.

```
{ id: "name"
, latitude: value
, longitude: value
}
```

5.6 Table

Tramite questo tipo di grafico è possibile creare delle tabelle con il numero desiderato di righe e colonne.

5.6.1 Table(title, headers, data, options)

Il metodo Table() permette di costruire una tabella, i parametri sono tutti obbligatori ad eccezione di options.

- **title**: prevede il passaggio di una stringa che rappresenterà il titolo del grafico;
- **headers**: prevede il passaggio di un array di stringhe che corrispondono ai titoli delle colonne della tabella. Il numero di colonne della tabella corrisponderà al numero di elementi di questo array;
- **data**: prevede il passaggio di un array di array, nel quale ogni array interno corrisponde ad una riga della tabella, come nell'esempio sottostante. L'array interno deve avere un numero di elementi corrispondente al numero di colonne altrimenti il programma terminerà.

$$[[x_{0,0}, y_{0,1}], [x_{1,0}, y_{1,1}], [x_{2,0}, y_{2,1}]]$$

Questo tipo di grafico può essere creato senza dati iniziali fornendo un array vuoto e inserendo i dati in un secondo momento tramite l'aggiornamento stream;

- **options**: prevede il passaggio di oggetto di tipo $JSON_G$ strutturato come nel seguente esempio. Non è necessario inserire tutti i parametri descritti sotto;

```
{ "insertPosition": "value"
, "orderBy":{ "column": 0
            , "order": "value"
            }
, "displayedLines": int
, "border": "value"
, "colorColumn" : []
, "colorRow" : []
, "colorCell" : []
, "colorColumnFont" : []
, "colorRowFont" : []
, "colorFont" : []
, "format": []
, "rowsLimit": 300
};
```

  - **insertPosition**: permette l'inserimento del valore top per inserire i nuovi dati all'inizio della tabella durante l'aggionamento stream del grafico, oppure bottom per inserire i dati dal fondo. Di default vengono inseriti in fondo;
  - **orderBy**: permette l'inserimento di un $JSON_G$ come sotto descritto per definire secondo quale colonna ordinare la tabella. Nel caso si scelga di utilizzare questa opzione tutte le opzioni nel $JSON_G$ sotto descritto devono essere inserite;

```
{ "column": value
, "order": "value"
}
```

    * **column**: prevede l'inserimento dell'indice della colonna rispetto alla quale si vuole ordinare la tabella, il minimo valore ammesso è 0;
    * **order**: prevede l'inserimento del valore ascending per ordinare la colonna in ordine crescente, o il valore descending per ordinare la colonna in ordine decrescente.
  - **displayedLines**: permette l'inserimento di un numero intero maggiore di 0, esso rappresenterà il numero di linee della tabella visualizzate nella pagina;
  - **border**: permette l'inserimento del valore true per visualizzare i bordi nella tabella, oppure false nel caso non si voglia visualizzarli. Di default i bordi vengono visualizzati;
  - **colorColumn**: permette di impostare il colore di sfondo di una o più colonne della tabella. Questo parametro prevede il passaggio di un array di array, dove ogni array interno corrisponde ad una colonna. L'array interno deve essere composto da due celle, nella prima deve essere inserito l'indice della

colonna e la seconda deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali.

```
[ [indiceColonna, "" ], ... ]
```

Nel caso non vengano inseriti colori, di default sarà impostato bianco da Norris;

- **colorRow**: permette l'inserimento del colore di sfondo di una o più righe della tabella. Questo parametro prevede il passaggio di un array di array, dove ogni array interno corrisponde ad una riga. L'array interno deve essere composto da due celle, nella prima deve essere inserito l'indice della riga e la seconda deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali.

```
[ [indiceRiga, "" ], ... ]
```

Nel caso non vengano inseriti colori, di default sarà impostato bianco da Norris;

- **colorCell**: permette l'inserimento del colore di sfondo di una cella della tabella. Questo parametro prevede il passaggio di un array di array, dove ogni array interno corrisponde ad una cella. L'array interno deve essere composto da tre celle nella prima deve essere inserito l'indice della riga, nella seconda deve essere inserito l'indice della colonna e la terza deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali.

```
[ [indiceRiga, indiceColonna, "" ], ... ]
```

Nel caso non vengano inseriti colori, di default sarà impostato bianco da Norris;

- **colorColumnFont**: permette di impostare il colore del testo di una o più colonne della tabella. Questo parametro prevede il passaggio di un array di array, dove ogni array interno corrisponde ad una colonna. L'array interno deve essere composto da due celle, nella prima deve essere inserito l'indice della colonna e la seconda deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali.

```
[ [indiceColonna, "" ], ... ]
```

Nel caso non vengano inseriti colori, di default sarà impostato nero da Norris;

- **colorRowFont**: permette l'inserimento del colore del testo di una o più righe della tabella. Questo parametro prevede il passaggio di un array di array, dove ogni array interno corrisponde ad una riga. L'array interno deve essere composto da due celle, nella prima deve essere inserito l'indice della riga e la seconda deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali.

```
[ [indiceRiga, "" ], ... ]
```

Nel caso non vengano inseriti colori, di default sarà impostato nero da Norris;

- **colorFont**: permette l'inserimento del colore del testo di una cella della tabella. Questo parametro prevede il passaggio di un array di array, dove

ogni array interno corrisponde ad una cella.
L'array interno deve essere composto da tre celle, nella prima deve essere inserito l'indice della riga, nella seconda deve essere inserito l'indice della colonna e la terza deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali.

```
[ [indiceRiga, indiceColonna, "" ], ... ]
```

Nel caso non vengano inseriti colori, di default sarà impostato nero da Norris;

- **format**: prevede il passaggio di un array di $JSON_G$, contenenti l'indice della colonna di riferimento, il tipo di valore, e il numero di decimali, per cambiare il formato dei dati di una colonna;

```
{ "column": value
, "valueType": "value"
, "decimals": value
}
```

  - **column**: prevede l'inserimento dell'indice della colonna rispetto alla quale si vuole cambiare il formato dei dati, il minimo valore ammesso è 0;
  - **valueType**: prevede l'inserimento dei valori euro, pounds o dollars per scegliere il formato dei dati visualizzati;
  - **decimals**: prevere l'inserimento di un valore compreso tra 0 e 6 per impostare il numero di cifre dopo la virgola che verranno visualizzate.
- **rowsLimit**: prevede il passaggio di un intero che permette di impostare il numero di dati che saranno mantenuti nel $client_G$, impostando cosi un upper bound. Il valore di default è impostato a 300.

Nel caso venga passato un dato non conforme alle specifiche il software terminerà.

5.6.2 getChartInfo()

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo $Table_G$, restituisce un oggetto $JSON_G$ contenente tutti i dati e i dettagli riguardanti il grafico.

5.6.3 updateInPlace(row, column, newValue, options)

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo $Table_G$, permette di aggiornare il valore di una cella già presente nella tabella. Tutti i parametri sono obbligatori, ad eccezione di options.

- **row**: permette l'inserimento di un numero intero corrispondente al numero di riga della cella nella quale si vuole modificare il dato. Il valore deve essere maggiore o uguale a 0, e inferiore al numero di righe della tabella. In caso non sia conforme a queste specifiche l'aggiornamento non avverrà;
- **column**: permette l'inserimento di un numero intero corrispondente al numero di colonna della cella nella quale si vuole modificare il dato. Il valore deve essere maggiore o uguale a 0, e inferiore al numero di colonne della tabella. In caso non sia conforme a queste specifiche l'aggiornamento non avverrà;

- **newValue**: prevede il passaggio del nuovo valore da inserire nella cella;
- **options**: prevede sia fornito un oggetto di tipo $JSON_G$ strutturato come nel seguente esempio. Tutti i parametri sono opzionali;

```
{ "colorFont": "value"
, "colorCell": "value"
}
```

  - **colorFont**: permette l'inserimento del colore del testo della cella. Deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali;
  - **colorCell**: permette l'inserimento del colore dello sfondo della cella. Deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali.

5.6.4 updateStream(data, options)

Questo metodo, invocato su un oggetto che rappresenta un grafico di tipo $Table_G$, permette l'inserimento di una nuova riga nella tabella. Il parametro data è obbligatorio, mentre options è opzionale.

- **data**: permette il passaggio di un array di dimensione pari al numero di colonne della tabella, l'inserimento verrà effettuato in testa o in coda in base a quanto specificato nel momento della creazione del grafico. Se la dimensione dell'array sarà inferiore al numero di colonne della tabella, le celle non specificate rimarranno vuote, mentre se la dimensione sarà maggiore del numero di colonne, le celle in eccesso non verranno considerate;
- **options**: prevede sia fornito un oggetto di tipo $JSON_G$ strutturato come nel seguente esempio. Tutti i parametri sono opzionali;

```
{ "colorRow": "value"
, "colorRowFont": "value"
, "colorCell": []
, "colorFont": []
};
```

  - **colorRow**: prevede sia fornito il colore di sfondo delle celle della riga inserita in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali;
  - **colorRowFont**: prevede sia fornito il colore del testo delle celle della riga inserita in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali;
  - **colorCell**: permette l'inserimento del colore di sfondo di una o più celle della riga inserita. Questo parametro prevede il passaggio di un array di array, dove ogni array interno corrisponde al valore per una cella. L'array interno deve essere composto da due celle, nella prima deve essere inserito l'indice della colonna e la seconda deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali.

```
[ [indiceColonna, "" ], ... ]
```

Nel caso l'indice specificato non corrisponda ad alcuna cella esistente, o nel caso non vengano inseriti colori, di default sarà impostato bianco da Norris;

– **colorFont**: permette l'inserimento del colore del testo di una o pià celle della riga inserita. Questo parametro prevede il passaggio di un array di array, dove ogni array interno corrisponde al valore per una cella. L'array interno deve essere composto da due celle, nella prima deve essere inserito l'indice della colonna e la seconda deve contenere il colore in formato $HTML_G$ esadecimale, ovvero il carattere # seguito da sei cifre esadecimali.

```
[ [indiceColonna, "" ], ... ]
```

Nel caso l'indice specificato non corrisponda ad alcuna cella esistente, o nel caso non vengano inseriti colori, di default sarà impostato nero da Norris.

A Glossario

A

API (Application Programming Interface)
Le API sono procedure disponibili al programmatore utili al compimento di un certo compito all'interno di un programma.

B

Bar Chart
Un grafico a barre è un confronto grafico di grandezze diverse, in cui la lunghezza delle barre orizzontali o verticali rappresentano la grandezza relativa dei valori.

C

Client
Un client, in informatica, indica una componente che accede ai servizi o alle risorse di un'altra componente detta *server*$_G$. In questo contesto si può quindi parlare di client riferendosi all'hardware oppure al software. Il termine client indica anche il software usato sul computer client per accedere alle funzionalità offerte dal *server*$_G$.

D

Double
Indica il metodo di rappresentazione approssimata dei numeri reali e di elaborazione dei dati usati dai processori per compiere operazioni matematiche.

E

Express
Framework$_G$ di *Node.js*$_G$ progettato per costruire applicazioni web in pagina singola, multipla, o ibride.

F

Framework
Un framework è un'architettura logica di supporto su cui un software può essere pro-

gettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.

H

HTML (HyperText Markup Language)
HTML in informatica è il linguaggio solitamente usato per la formattazione di documenti ipertestuali disponibili nel World Wide Web sotto forma di pagine web.

HTTP (HyperText Transfer Protocol) L'HyperText Transfer Protocol (protocollo di trasferimento di un ipertesto) è usato come principale sistema per la trasmissione d'informazioni sul web, ovvero in un'architettura tipica client-server. Le specifiche del protocollo sono gestite dal World Wide Web Consortium (W3C). Un $server_G$ HTTP generalmente resta in ascolto delle richieste dei $client_G$ sulla porta 80 usando il protocollo TCP a livello di trasporto.

J

JSON (JavaScript Object Notation)
JSON è un formato standard per lo scambio di dati in applicazioni $client_G$-$server_G$. I dati vengono rappresentati tramite la coppia attributo-valore.

L

Line Chart
Un grafico a linee o grafico lineare è un tipo di grafico, che mostra le informazioni come una serie di punti di dati connessi da segmenti di linea retta. Si tratta di un'estensione di un grafico a dispersione, e viene creata collegando una serie di punti che rappresentano singole misurazioni con i segmenti di linea.

M

Map Chart
Il grafico Map Chart è una mappa geografica con dati rappresentati in sovraimpressione. I dati possono essere rappresentati come forme geometriche colorate, come icone o come testo. Generalmente le forme geometriche e le icone possono essere selezionabili, cosi come la mappa può essere ingrandita o rimpicciolita a discrezione dell'utente. Le rappresentazioni dei dati cambiano contestualmente alla porzione di mappa visualizzata.

Marker
Un marker è un simbolo visualizzato in un grafico di tipo *Map Chart*$_G$ che serve ad identificare un singolo punto della mappa.

Middleware
Insieme di programmi informatici che fungono da intermediari tra diverse applicazioni e componenti software. Sono spesso utilizzati come supporto per sistemi distribuiti complessi.

Mount
É un comando dei sistemi operativi Unix che permette il montaggio di un file system (di un dispositivo esterno o di un'altra partizione del disco rigido) agganciandolo a una directory del file system in uso, in modo da rendere accessibili ai programmi e agli utenti del sistema file e directory in esso contenuti.
Nel caso di Norris, viene inteso come il rendere disponibile agli utenti finali una risorsa pagina ad un determinato indirizzo URL dell'applicazione.

N

Namespace
In informatica è una collezione di nomi di entità, definite dal programmatore, omogeneamente usate in uno o più file sorgente. Lo scopo dei namespace è quello di evitare confusione ed equivoci nel caso siano necessarie molte entità con nomi simili, fornendo il modo di raggruppare i nomi per categorie.
Nel caso di Norris, serve per evitare confusione tra l'istanza di socket.io principale dello sviluppatore e quella necessaria per il funzionamento del *framework*$_G$.

Node.js
Framework$_G$ open source per applicazioni *server*$_G$-side e di rete. Le applicazioni Node.js sono scritte in Javascript. Node.js fornisce un'architettura event-driven e una *API*$_G$ I/O non bloccante che ottimizza la capacità di trasmissione e la scalabilità di un'applicazione.

npm
É il package manager ufficiale per *Node.js*$_G$ Gestisce le dipendenze e permette di installare le applicazioni registrate nel registro npm. Viene eseguito tramite il comando npm.

P

Push
Push, o Server Push, è un metodo di comunicazione di rete dove l'informazione viene inviata ad un *client*$_G$ su iniziativa del *server*$_G$, senza che sia necessaria una richiesta del *client*$_G$.

R

Routing
Il routing è l'instradamento effettuato a livello di rete. L'instradamento, nel campo delle reti di telecomunicazione, è la funzione di un commutatore (router nel caso di Norris) che decide su quale porta o interfaccia inviare un elemento di comunicazione ricevuto (conversazione telefonica, pacchetto dati, cella, flusso di dati).

S

SDK
In informatica, indica genericamente un insieme di strumenti per lo sviluppo e la documentazione di software.
Molti SDK sono disponibili gratuitamente e possono essere prelevati direttamente dal sito del produttore: in questo modo si cerca di invogliare i programmatori ad utilizzare un determinato linguaggio o sistema. Vi è anche un utilizzo orientato al mercato: alcuni programmi vengono venduti assieme al loro SDK permettendo ai compratori di sviluppare ulteriori parti del programma comprato.

Server
É un componente o sottosistema informatico di elaborazione che fornisce, a livello logico e a livello fisico, un qualunque tipo di servizio ad altre componenti ,tipicamente chiamate *client*$_G$. La richiesta di questi servizi da parte dei *client*$_G$ avviene attraverso una rete di computer, all'interno di un sistema informatico o direttamente in locale su un computer.

Socket.io
Libreria Javascript che permette comunicazione bidirezionale tra *client*$_G$ e *server*$_G$. Utilizza il protocollo WebSocket.

T

Table
Il grafico di tipo Table è una griglia sulla quale vengono visualizzati i dati.

B Contatti

Se riscontrate problemi o bug durante l'installazione o l'utilizzo del *framework*$_G$ Norris potete segnalare tali anomalie tramite l'apertura di una issue di GitHub al seguente indirizzo:

https://github.com/FlameTech/Norris-rtbi/issues

Dopo aver effettuato l'operazione di login vi troverete nella pagina visualizzata nella Figura 1. A questo punto è sufficiente selezionare il pulsante New issue per creare una segnalazione

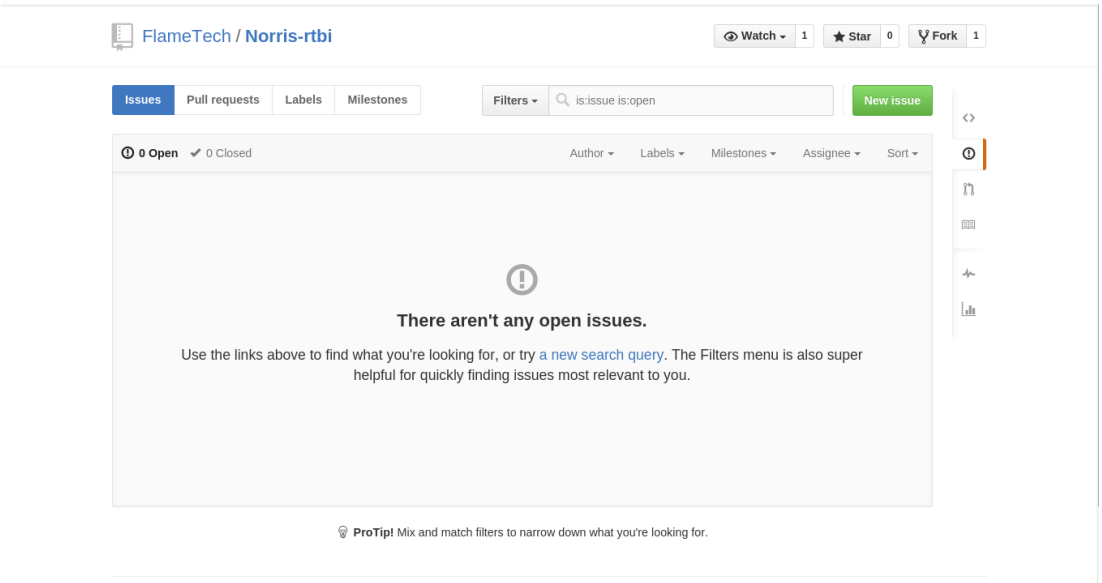

Figura 1: Sezione issue

Dopo che avete compilato i campi basta premere il pulsante Submit new issue per inoltrare la segnalazione

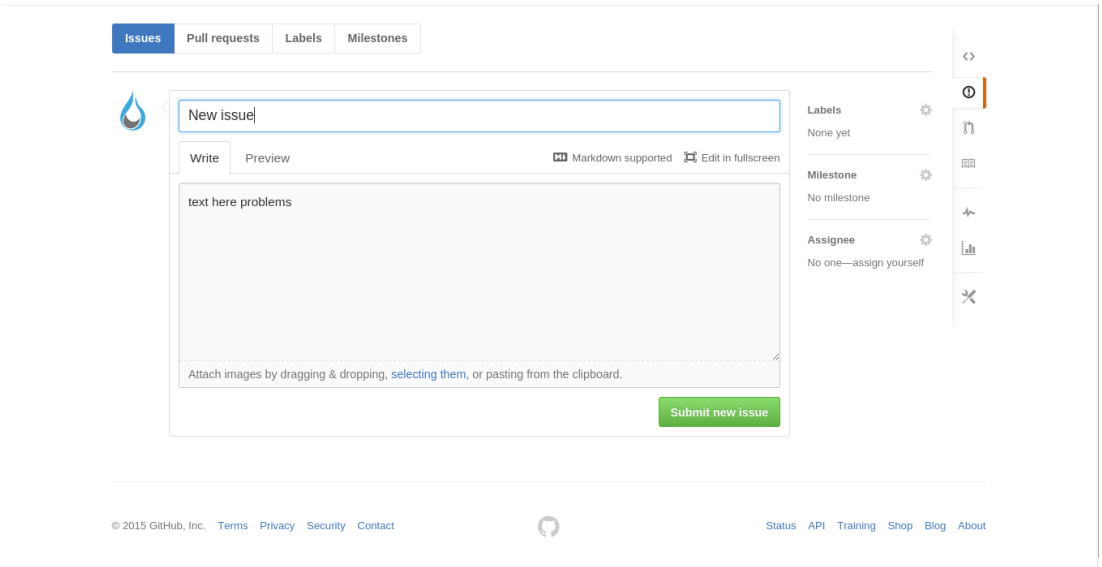

Figura 2: Creazione di un issue