

Wizard Yaml Configurator

Matteo Menghini

Sommario

Wizard Yaml Configurator	1
1. Introduzione	2
2. Struttura dell'applicazione	2
2.1. Top Bar.....	3
2.2. General	4
2.3. Views	4
2.4. Data	5
3. YAML.....	7
4. Implementazione.....	9
4.1. Struttura del codice	9
4.2. Controlli e gestione errori	10
5. Utilizzo e Testing.....	10

1. Introduzione

Wizard Yaml Configurator(WYC) è un applicazione che permette a chi lo utilizza di creare una configurazione YAML per i generatori del progetto MMI4.0 . WYC mette a disposizione una serie di campi da compilare o da scegliere un valore, i quali, se compilati correttamente, daranno origine ad una configurazione in formato .yaml che potrà essere utilizzata dai generatori per produrre le interfacce grafiche e/o a linguaggio naturale.

WYC è scritto in Javascript utilizzando la libreria React.js con il supporto dei componenti di PrimeReact.

2. Struttura dell'applicazione

L'applicazione consiste in un processo guidato nella compilazione di campi che aiuteranno l'utente a decidere come sarà la struttura della propria dashboard. La struttura dell'applicazione si può suddividere in quattro sezioni: TopBar, General , Views e Data.

2.1. Top Bar



Fig.1 TopBar

La top bar è la barra posizionata in alto nell'applicazione e sarà sempre visibile per permettere all'utente di accedere alle tre funzioni che mette a disposizione. Queste tre funzioni che sono accessibili tramite dei bottoni all'interno della top bar e sono:

- **Download:** Permette di scaricare il file .yaml in base ai dati compilati fino a quel momento. Si può scaricare un file anche senza aver compilato niente, il quale sarà semi-vuoto. Questo file vuoto se passato ai generatori restituirà una dashboard vuota e/o un bot senza interazioni.
- **Upload:** Permette di caricare un file .yaml nell'applicazione. Se il file ha una struttura conforme con quella definita dal progetto MMI40(Vedi capitolo YAML) allora verranno visualizzati tutti i campi già compilati (nello yaml) all'interno dell'applicazione. Se il file non è un .yaml verrà restituito un messaggio di errore.
- **Algoritmo:** Premendo il tasto "Algoritmo" si aprirà un menu laterale con delle domande e un menu a tendina con alcune possibili risposte. L' algoritmo consiste in una procedura guidata che permette all'utente di scegliere varie risposte a delle domande mirate sulle informazioni del dato che si vuole visualizzare. Il risultato sarà un tipo di grafico che si dovrebbe utilizzare per visualizzare meglio il tipo di dato scelto durante l'algoritmo.

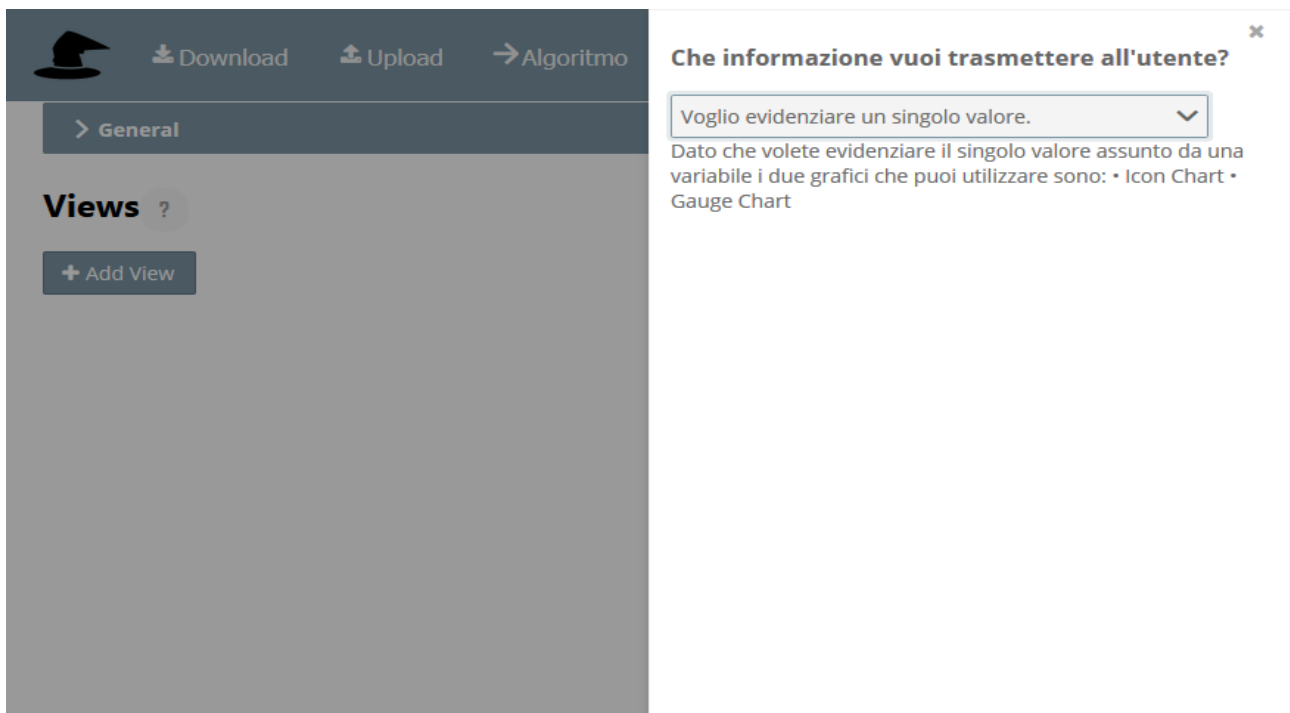
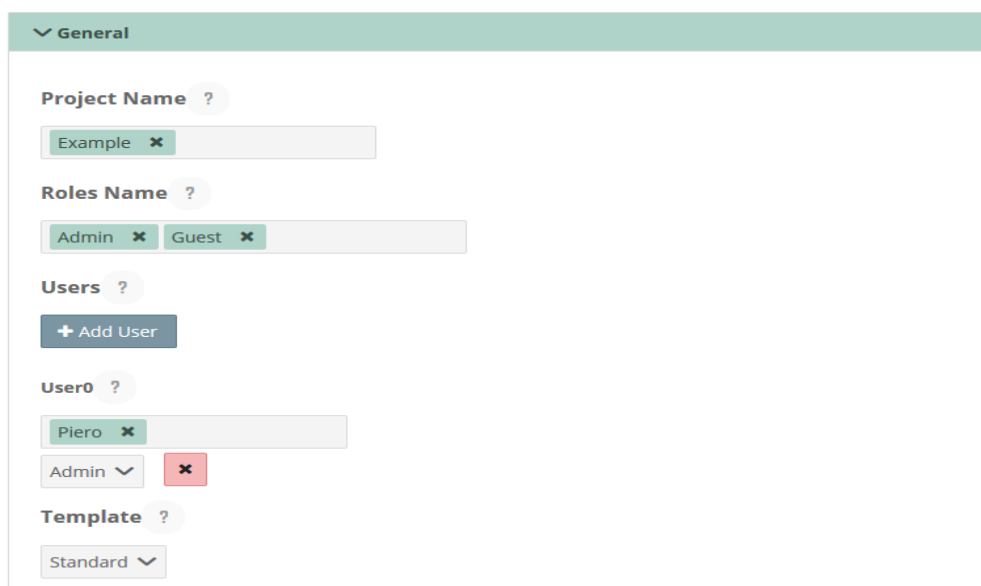


Fig.2 Sezione dell'algoritmo

2.2. General

La sezione General contiene i campi da compilare relativi alle informazioni generali della dashboard. Questi campi sono:

- **Project Name:** Nome generale del progetto. Sarà il nome della dashboard e del bot. Bisogna inserire il valore e premere Enter per confermarlo.
- **Roles:** Lista dei nomi dei ruoli che si vogliono all'interno della dashboard. Servono a limitare gli accessi ad alcuni dati.
- **Users:** Coppia <nome utente, ruolo>. Corrispondono agli utenti iniziali che avranno accesso alla dashboard. Il ruolo va scelto tra i valori inseriti nel campo Roles.
- **Template:** Il template utilizzato dalla dashboard (per adesso solo "Standard").



The screenshot shows the 'General' settings section with a teal header. It contains several form fields: 'Project Name' with a text input containing 'Example'; 'Roles Name' with a multi-select input containing 'Admin' and 'Guest'; 'Users' with a '+ Add User' button; 'User0' with a text input containing 'Piero' and a dropdown menu currently set to 'Admin'; and 'Template' with a dropdown menu set to 'Standard'. Each field has a help icon (?) to its right.

Fig.3 Sezione General

2.3. Views



The screenshot shows the 'Views' section with a teal header. It features a '+ Add View' button and a list of three views: 'Sala Caldaie', 'Centrale Elettrica', and 'Vivaio'. Each view entry has a right-pointing chevron icon (>) on the left. A mouse cursor is visible over the 'Views' header.

Fig 4 Sezione Esterna View.

La sezione Views può essere vista come una lista delle pagine che andranno a comporre la dashboard ed è composta da un bottone “Add View” per aggiungere una pagina e dalla lista delle view.

Ogni view contiene i seguenti campi:

- **View Name:** Nome della pagina. Deve essere univoco tra le view.
- **View Role:** Lista dei ruoli che hanno accesso a quella view. La lista viene presa dal campo Roles presente nella sezione General.
- **Delete Button:** Bottone per eliminare la view corrente.
- **Data:** Sezione data della view.



Fig.5 Panoramica con una vista aperta.

2.4. Data

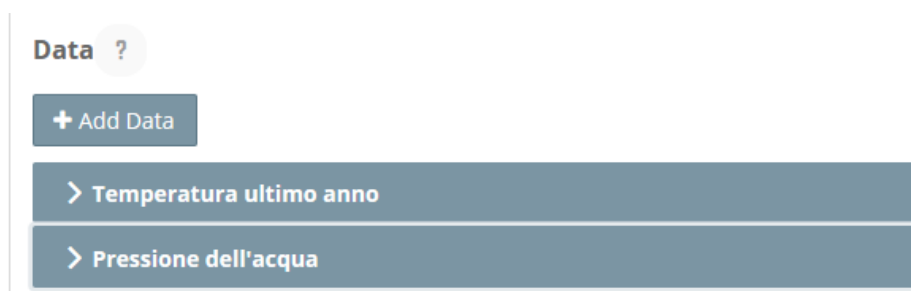


Fig. 6 Sezione Data.

La sezione Data è una lista dei dati contenuti in una view con le relative informazioni riguardanti quel dato. Un nuovo dato può essere creato premendo il bottone “Add Data” ed ogni dato contiene i seguenti campi:

- **Title:** Nome del dato, deve essere univoco all’interno della view.
- **Type:** Tipo di dato, serve per segnalare l’unità di misura da usare.
- **Device:** Lista dei device dove prendere i dati.
- **Keyword:** Lista dei tag riguardanti il dato.
- **Alarm:** Valore booleano per definire se al dato è associato un allarme.
- **Max Threshold:** Valore massimo che deve essere superato per far partire l’allarme (solo se Alarm è attivo).
- **Min Threshold:** Valore minimo che se è al di sotto scatta l’allarme (solo se Alarm è attivo).
- **Aggregation Type:** Tipo di funzione di aggregazione da usare per il dato. Valori possibili -Min , Sum, Max, Mean.
- **Aggregation Partition:** Valore su quali dati aggregare tra tag e device. Può essere scelto solo un valore. (Per maggiori dettagli consultare il documento charts.pdf nella sezione 4.4.3)
- **Time Interval:** Intervallo di tempo che si vuole mostrare.
- **Granularity:** Numero di punti da rappresentare nel grafico.
- **Charts:** tipo di chart da utilizzare per mostrare il grafico.

The screenshot shows a web interface for adding data. At the top, there's a navigation bar with a hat icon, 'Download', 'Upload', and 'Algoritmo' buttons. Below this is a '+ Add Data' button. The main form is titled 'Temperatura ultimo anno' and has a red 'X' button in the top right corner. The form contains four sections: 'Title' with a text input field containing 'Temperatura ultimo anno'; 'Type' with a dropdown menu showing 'temperature'; 'Device' with a text input field containing two tags: 'Device1' and 'Device2'; and 'Keyword' with a text input field containing a tag 'Temp' and a button 'Select a Keyword'.

Fig.7 Panoramica parziale con aperto un dato

Ogni chart ha delle proprietà associate e i campi associati a queste proprietà si mostrano quando si è scelto il chart. Tutti i campi possibili sono:

- **Legend Position:** Dove posizionare la legenda del grafico. Valori possibili Top, Bottom, Right, Left e None.
- **Trend Line:** Valore booleano se si vuole mostrare la trend line dello sciame dei punti.
- **Min Value:** valore che rappresenta lo 0% del grafico. Usato nei grafici Compass e Gauge.
- **Max Value:** Valore che rappresenta il 100% del grafico. Usato nei grafici Compass e Gauge.
- **Start Angle:** Angolo rispetto all'asse y dove inizia il grafico (valori consentiti tra -180 e 0). Usato nei grafici Compass e Gauge.
- **End Angle:** Angolo rispetto all'asse y dove finisce il grafico (valori consentiti tra 0 e 180). Usato nei grafici Compass e Gauge.
- **Name Icon:** Nome del file dove prendere l'icona per il grafico di tipo Icon.
- **Value Position:** Dove mettere il valore rispetto all'icona del Icon Chart. Valori possibili Top, Bottom, Right, Left.
- **Line Type:** Tipo di linea per il grafico di tipo Line . Valori possibili -Point, Basic, Brush.

Charts

Type ?

Compass ▾

Min Value ?

-50 ▴ ▾

Max Value ?

175 ▴ ▾

Start Angle ?

-90 ▴ ▾

End Angle ?

180 ▴ ▾

Fig.8 Sezione del chart Compass con le relative proprietà.

3. YAML

Dopo aver compilato i campi necessari l'utente può premere Download e scaricare il file. Questo file in formato .yaml conterrà tutte le informazioni inserite all'interno del programma in modo molto simile ai campi compilati. Come si può vedere dalla struttura dello yaml in figura 9 se si compila il campo Project Name nell'applicazione con la parola "example", nello yaml scaricato risulterà compilato il campo "projectname: example".

```

projectname: "Example"
template: "Standard"
roles:
  - "Admin"
  - "Guest"
views:
  - viewname: "Sala Caldaie"
    admittedroles:
      - "Admin"
      - "Guest"
    data:
      - title: "Temperatura ultimo anno"
        type: "temperature"
        alarm:
          maxthreshold: 120
          minthreshold: -15
        variables:
          datasource:
            device:
              - "Device1"
              - "Device2"
            keyword:
              - "Temp"
          aggregationfunction:
            type: "mean"
            aggregated:
              - "device"
              - "keyword"
            divided: []
        timeinterval: "1 years"
        granularity: "1 months"
        chart:
          type: "line"
          legendPosition: "bottom"
          lineType: "basic"

```

Fig 9 Parte iniziale del file yaml

Una nota aggiuntiva va fatta sul campo "users" e sul campo "type" all'interno della sezione data; nel primo(users) va inserito il nome dell'utente e il ruolo ma, giustamente, per accedere alle dashboard finali il nome utente non basta, serve anche una password. L'applicazione quando genera un nuovo utente crea anche una password univoca a 8 caratteri che si troverà anche nello yaml (figura 10), la quale password potrà essere utilizzata per il primo accesso.

Il campo "type" invece permette di scegliere il tipo di dato che si vuole visualizzare ma non permette di scegliere l'unità di misura; però quando viene generato il file .yaml nella parte finale c'è una lista dei tipi di dato con un'unità di misura standard associata. Esempio: l'unità di misura standard dell'applicazione per la temperatura è il Celsius ma se l'utente nella sua dashboard vuole visualizzare la temperatura in kelvin dovrà modificare a mano nello yaml il campo.


```

users:
  - name: "Piero"
    pass: "mdBfpAlf"
    role: "Admin"
  - name: "Franco"
    pass: "mhvg40un"
    role: "Guest"
datatype:
  - name: "wind direction"
    unitofmeasure: "°"
  - name: "temperature"
    unitofmeasure: "°C"
  - name: "flow rate"
    unitofmeasure: "m ^ 3 / s"
  - name: "pumping speed"
    unitofmeasure: "l / s"
  - name: "speed"
    unitofmeasure: "m / s"
  - name: "acceleration"

```

Fig. 10 Parte finale dello yaml

4. Implementazione

Come già detto l'applicazione è scritta in Javascript utilizzando React.js. Inizialmente è stata pensata come una pagina web statica accessibile tramite un URL però in seguito ho pensato che forse un applicazione desktop potesse tornare più semplice. Alla fine, ho optato per entrambe. Visto che il codice è accessibile da la possibilità di creare sia la pagina che la app desktop. In aggiunta alle tecnologie citate prima aggiungo anche Electron, un framework che permette di convertire pagine web in applicazioni desktop.

4.1. Struttura del codice

Per facilitare la lettura del codice la struttura è stata suddivisa in componenti minori. Il progetto non presenta funzioni complesse, la maggior parte di queste sono solo assegnamenti del valore scelto nel campo allo stato. I file presenti nel codice sono:

- **Index.js:** contiene il render principale dell'intera struttura dell'applicazione.
- **App.js:** contiene la struttura e anche tutte le variabili dei vari campi. Tutti i campi sono coppie di componenti, il primo è un NameComp(vedi NameComp.js) e il secondo è un componente di PrimeReact che può essere un Button, un InputText, un DropDown o altro.
- **Component/TopBar:** reindirizza la struttura della TopBar.
- **Component/Data:** reindirizza la struttura della sezione Data.
- **Component/View:** reindirizza la struttura della sezione View.

- **Component/NameComp:** reindirizza l'header e il tips dei campi. Crea un componente formato da un titolo del campo sottostante e da una icona che se cliccata dà informazioni sul campo.
- **Component/ChartProps:** reindirizza i campi delle proprietà relativi al chart scelto.
- **AlgChart.js:** contiene la struttura dell'algoritmo della scelta del chart.
- **Function/:** contiene varie funzioni utilizzate dall'applicazione.
- **Main.js:** File che contiene le informazioni per creare l'applicazione desktop del programma.


4.2. Controlli e gestione errori

Non ci sono particolari controlli nel programma; c'è un controllo sui nomi delle view, degli user e dei data che se vengono inseriti nomi già usati compare un messaggio di errore. Un altro controllo viene fatto sul tipo di file caricato tramite upload, se il file non è una yaml viene mostrato un messaggio di errore. Se il file è una yaml vengono controllati i campi per vedere se è uno yaml corretto. Non tutti i campi sono controllati completamente, se si devono modificare dei valori a mano (cambiare String con Int) si deve fare dopo aver generato lo yaml e sarebbe inopportuno ripassarlo all'applicazione tramite upload poiché potrebbe mandare in crash il programma.

5. Utilizzo e Testing

Il codice è facilmente reperibile su GitHub al link <https://github.com/WilliamSimoni/MMI40/tree/wizard2.0> però per testarlo vanno eseguiti un paio di passaggi:

1. Bisogna installare sul proprio dispositivo Node.js (un runtime di javascript). Durante lo sviluppo ho utilizzato la versione v12.13.1. Dopo averlo installato controllare se tutto è andato bene aprendo il terminale e digitando i comandi "node --version" e "npm --version" (npm è il gestore di pacchetti).

 Prompt dei comandi

```
C:\>node --version
v12.13.1

C:\>npm --version
6.12.1

C:\>
```

2. Sempre con il terminale (prompt dei comandi) andare nella directory del codice dell'applicazione (scaricata dal link del github sopra). Nella directory ci devono essere i file "package.json", "package-lock.json" e le cartelle "src" e "public". Fatto ciò nel terminale digitare il comando "npm install" il quale installerà tutti i pacchetti necessari per far funzionare l'applicazione (richiede un po' di tempo in base al dispositivo che si utilizza).

Se si usa Visual Studio Code come editor di codice si può aprire la directory tramite Visual studio e poi utilizzare il terminale messo a disposizione.

3. Adesso si dispone dei pacchetti necessari testare l'applicazione come pagina web. Sempre con il terminale digitare "npm start" dopo qualche secondo si aprirà il browser all'indirizzo <http://localhost:3000/> e si potrà testare l'applicazione. Per interromperla nel terminale premere "CTRL + C" e dare conferma.
4. Se si vuole testare l'applicazione tramite app desktop bisogna eseguire qualche passaggio in più. Per primo se npm start è attivo bisogna interromperlo, poi nel terminale (sempre dentro la directory) digitare "npm run build" che crea la build del progetto. Dopo che il comando ha finito si può digitare "npm run electron" che ci farà testare l'applicazione come app desktop.
5. Infine, se si vuole creare l'eseguibile dell'applicazione bisogna digitare (sempre nel terminale) "npm package-win". Questo comando crea solo il package con l'eseguibile per windows e sarà dentro la cartella "release-builds", mentre se si vuole creare il package per linux bisogna eseguire "npm package-linux" o "npm package-mac" per MacOS.