



**Università  
degli Studi  
di Ferrara**

## **Corso di laurea in Ingegneria Elettronica e Informatica**

Piattaforma E-learning per scuole superiori:  
interfaccia responsive per tabelle web su  
dispositivi mobili.

Relatrice:

**Prof. Elena Bellodi**

Laureando:

**Sebastiano Cavani**

Secondo relatore:

**Dr.-Ing Nguembang Fadja Arnaud**

Anno accademico 2022/2023



# Indice

<b>Introduzione</b>	<b>5</b>
<b>1 Scopo del progetto</b>	<b>7</b>
<b>2 Tecnologie utilizzate</b>	<b>10</b>
<b>3 Implementazione delle tabelle</b>	<b>12</b>
3.1 Analisi della struttura già esistente delle tabelle . . . . .	12
3.2 Modifiche per rendere l'interfaccia responsive . . . . .	23
<b>Conclusione</b>	<b>42</b>
<b>Sitografia</b>	<b>44</b>
<b>Ringraziamenti</b>	<b>46</b>



## Introduzione

E-learning è la piattaforma educativa in corso di sviluppo da SYSAIT (System Afrik Information Technology) pensata per migliorare e digitalizzare la gestione del sistema scolastico africano. Il progetto affrontato in questa tesi fornisce un contributo a tale piattaforma, ponendosi come obiettivo quello di sviluppare un'interfaccia responsive per le tabelle non adattabili a dispositivi mobili utilizzate all'interno dell'applicazione.

Tale attività di tesi è iniziata quando SYSAIT aveva già sviluppato un componente funzionante per la visualizzazione dei dati in formato tabellare, pertanto oltre a descrivere le modifiche effettuate per raggiungere l'obiettivo del progetto ci si occuperà anche di analizzare l'implementazione adatta solamente per la navigazione da computer. Il componente oggetto di intervento è impiegato frequentemente nel sistema per presentare in modo organizzato, ad esempio, le informazioni sulle scuole registrate nella piattaforma o i dati relativi agli studenti iscritti in un determinato istituto.

Per portare a termine il lavoro sono state sfruttate alcune conoscenze di base del corso di Ingegneria dei Sistemi Web, alle quali è stato necessario aggiungere nuove competenze fondamentali acquisite durante la fase iniziale del progetto. Queste riguardano il framework Quasar, basato sul framework javascript Vue.js, e l'utilizzo di nuovi strumenti come Confluence e Bitbucket.



# 1 Scopo del progetto

Prima di parlare dell'obiettivo specifico del lavoro svolto è importante descrivere le origini e il contesto in cui ci si trova.

SYSAIT è una giovane azienda nata con la visione di migliorare la vita di tutti i giorni grazie alla digitalizzazione. L'obiettivo principale è quello di creare soluzioni innovative che possano avere un significativo impatto sull'economia mondiale, in particolar modo su quella africana. L'idea alla base è quella di sviluppare prodotti digitali per migliorare tutte le infrastrutture, compreso il sistema amministrativo africano, il sistema sanitario e il sistema educativo. SYSAIT crede fortemente che la digitalizzazione sia la strada da seguire verso il successo per l'Africa, contribuendo a raccogliere i dati necessari per poter in futuro costruire sistemi basati sull'intelligenza artificiale.

Per il miglioramento del sistema educativo SYSAIT sta sviluppando E-learning, la piattaforma al centro di questa tesi. Essa potrebbe rappresentare una svolta nella gestione delle scuole superiori africane, dove l'archiviazione dei dati è ancora fatta manualmente, la gestione e pianificazione delle attività è molto onerosa, ed è difficile stabilire un rapporto diretto e costante tra insegnanti e genitori. Questi problemi e non solo rendono molto difficile creare un sistema educativo efficiente.

Parlando più nello specifico di E-learning, essa è la prima e unica piattaforma globale interamente cloud pensata per il settore, rappresentando quindi un riferimento funzionale, tecnologico e qualitativo per quest'ambito. La piattaforma è pensata per la gestione di oltre 10.000 studenti e 5.000 operatori, inoltre è multilingua (3 lingue supportate: francese, inglese, italiano), multistruttura e multiservizio. Tutte queste funzionalità hanno permesso a SYSAIT di acquisire oltre 100 clienti, anche di grandi dimensioni, a cui sono garantiti servizi molto diversi tra loro, come ad esempio la completa gestione dell'anagrafica, delle lezioni, delle iscrizioni, dei pagamenti delle tasse e molto altro.

La piattaforma si può suddividere in 3 diversi moduli:

- E-learning: serve per la gestione della scuola (modulo in comune a tutte le scuole registrate).
- E-school: sito di vetrina per ogni scuola (modulo distinto per ogni scuola).
- E-learning-api: API back-end dei due moduli precedenti.

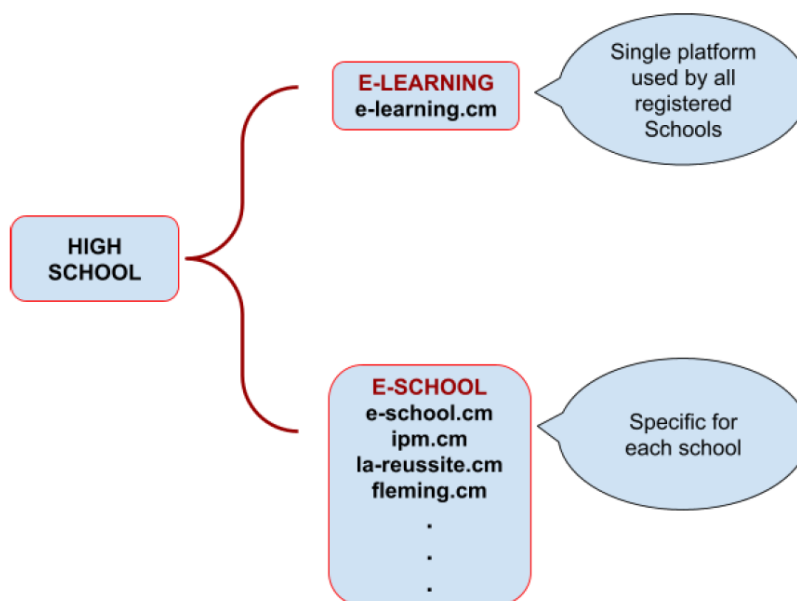


Figura 1.1

Per ogni scuola quindi è fornito sia un sito di vetrina che un'applicazione di gestione come si può vedere nella figura 1.1.

Il modulo E-school fornisce una presentazione della scuola in tutti i suoi aspetti, come ad esempio l'organizzazione della scuola stessa, informazioni sulle attività e sui club offerti agli studenti, la lista completa dei contatti con cui si può interagire con la scuola, l'indirizzo della sede, le ultime notizie, la sua storia, gli esami ufficiali, statistiche e molto altro. È incluso inoltre un CMS (Content Management System) per aiutare le scuole a creare, gestire o modificare tutte le funzioni descritte precedentemente.

E-learning invece ha una struttura un po' più complessa suddivisa in molteplici moduli:

- Modulo per gestire l'autenticazione di amministratori, insegnanti, studenti o genitori.
- Modulo per gestire il personale
- Modulo per gestire gli studenti
- Modulo per gestire le classi
- Modulo per gestire il programma scolastico
- Modulo per la gestione delle presenze/assenze degli studenti
- Modulo per la gestione dei compiti assegnati per casa
- Modulo per la gestione delle competizioni
- Modulo per la gestione degli esami (sequenziali e ufficiali)



- Modulo per la gestione del pagamento delle tasse
- Modulo per la gestione delle pagelle

L'obiettivo di questa tesi si concentra sullo sviluppo di un'interfaccia responsive per tabelle web non adattabili ai dispositivi mobili, cercando di ottimizzarne la visualizzazione migliorando l'esperienza utente e la leggibilità dei dati. Queste proprietà sono ormai fondamentali per ogni applicazione, dato il costante aumento di dispositivi con diverse caratteristiche con cui l'utente potrebbe accedere alla piattaforma.

Denominazione	Email	Indirizzo	Tutti i telefoni
 <b>IPM (SYSAT DIGITAL SCHOOL)</b> <small>80107370903</small> <small>ISTRUZIONE TECNICA E PROFESSIONALE(SOTTOSISTEMA FRANCESE)</small>	institutpaulmomo@gmail.com	Total Ebom, à 400 Mètre du Rond Point Damas en direction de la barrière, juste avant la, Yaoundé, Camerun	/00237698565158/00237698565158/0023 7698565158/00237237698565158
 <b>SYSAT DIGITAL SCHOOL (SYSAT DIGITAL SCHOOL)</b> <small>00000000000</small> <small>EDUCAZIONE GENERALE(SOTTOSISTEMA INGLESE)</small>	syaatechnology@gmail.com	Via Antonio Fortunato Oroboni, 80, 44122 Ferrara FE	/00393271852672/+49 015206657219/+49 015206657219/00393271852672

Figura 1.2: Tabella che rappresenta l'elenco delle scuole iscritte alla piattaforma

Il progetto in questione è stato sviluppato front-end ed è nativamente integrabile all'implementazione per le tabelle già presente in E-learning, la quale sarà soggetta di analisi prima di descrivere i cambiamenti necessari per raggiungere gli obiettivi prefissati.

Nello sviluppo dell'interfaccia responsive dovrà essere mantenuta la capacità di ordinare o filtrare le tabelle, funzione già disponibile nell'implementazione visibile nella figura 1.2. Inoltre verranno sfruttate alcune proprietà messe a disposizione dal framework Quasar per adattare la visualizzazione delle tabelle su schermi di varie dimensioni, applicando le necessarie modifiche a livello di design e di struttura del componente.

## 2 Tecnologie utilizzate

Lo sviluppo front-end della piattaforma, oltre ad utilizzare HTML, CSS e Javascript, è realizzato grazie al framework Quasar, il quale è basato sul framework javascript Vue.js.

Vue.js è un progressive framework open-source pubblicato nel febbraio del 2014 e pensato per la creazione di interfacce utente o, come in questo caso, di SPA (Single-Page Application). Questo termine è usato per indicare un'applicazione o un sito web che carica solamente una pagina dal server, evitando quindi di doverne caricare una nuova per ogni interazione. Le risorse sono caricate e aggiunte alla pagina solamente quando è necessario grazie ad una comunicazione dinamica back-end con il web server. Sviluppare un'applicazione in questo modo garantisce un guadagno in termini di performance e di reattività del sito.

Quasar è il framework open-source numero uno basato su Vue.js con il quale è possibile creare rapidamente applicazioni o siti web responsive. Il motto di Quasar è: "write code once and simultaneously depoly it" (scrivi il codice una volta e allo stesso tempo distribuisilo); questo framework infatti permette di creare applicazioni con una singola base di codice sia per la versione web che per la versione mobile, riducendo i tempi di sviluppo. Un altro vantaggio nell'utilizzare Quasar è il non essere obbligati ad usare librerie aggiuntive; ciò è possibile grazie a Quasar CLI, parte centrale del framework, che tramite un'implementazione leggera mette a disposizione tutti gli strumenti necessari per lo sviluppo front-end di un'applicazione.

Vue.js è integrato all'interno del framework Quasar grazie ai cosiddetti Single File Components, ovvero file con estensione .vue facilmente riutilizzabili in più parti dell'applicazione e dotati di una struttura ben precisa:

- Template: sezione corrispondente al body di un normale file Html, serve per definire come verrà visualizzata la pagina; grazie a diverse classi CSS messe a disposizione direttamente da Quasar e alle direttive offerte da Vue è possibile impostare un layout ordinato e reattivo in maniera relativamente semplice.
- Script: sezione dedicata al codice javascript, qui è possibile definire diversi aspetti del componente Vue, come ad esempio i dati a sua disposizione e i metodi per manipolarli; all'interno di questa parte del codice è possibile anche importare componenti provenienti da altri file .vue per poterli poi utilizzare nel template.
- Style: sezione opzionale per aggiungere classi CSS create dallo sviluppatore o per modificare alcuni aspetti delle classi offerte direttamente da Quasar.

È importante specificare che per la realizzazione dell'applicazione si sia scelto di utilizzare la versione 1.22.10 di Quasar, dotata della versione 2 di Vue.js, nonostante sia disponibile da giugno 2021 la versione 2 di Quasar, dotata invece della versione 3 di Vue.js.

Il front-end dell'applicazione comunica via API con il back-end, il quale è sviluppato con il framework open source Ruby on Rails. Inoltre la piattaforma si appoggia su un'unica base di dati PostgreSQL.

PostgreSQL è un potente DBMS (Data Base Management System, ovvero "Sistema di gestione di basi di dati") pubblicato nel 1986 e open source. Un DBMS è un sistema software che facilita il processo di definire, costruire, manipolare e condividere basi di dati. PostgreSQL è un DBMS relazionale, ovvero basato sul modello relazionale, che utilizza ed estende il linguaggio SQL combinato con numerose funzionalità per memorizzare in modo sicuro i dati e scalare i carichi di lavoro più onerosi.

Tutta la piattaforma è supportata quindi da un solo server back-end ed un'unica piattaforma di gestione per tutto il sistema. La presentazione delle scuole è gestita tramite numerosi applicativi, con ambiente di test su Heroku, *platform as a service* (PaaS) sul cloud, e AWS (Amazon Web Services) impiegato per la produzione.

Durante la fase iniziale di formazione si è sfruttata Confluence, una piattaforma collaborativa sviluppata da Atlassian dove è elencata la documentazione necessaria per poter sfruttare le tecnologie front-end usate per lo sviluppo di E-learning e molto altro. Attraverso Confluence è stato consigliato di lavorare in ambiente Linux per facilitare la gestione del progetto, per questo scopo è stato necessario creare una macchina virtuale tramite Oracle VM VirtualBox. Su Confluence sono fornite anche le istruzioni per la clonazione del progetto esistente, modificabile poi in locale utilizzando come editor Visual Studio Code.

Il codice vero e proprio della parte front-end dell'applicazione è stato reso disponibile tramite Bitbucket, una piattaforma di proprietà di Atlassian che offre il servizio di git repository. Bitbucket è stato molto utile per tenere traccia dei cambiamenti effettuati al progetto iniziale, per favorire una comunicazione costante su eventuali modifiche da apportare in corso d'opera e per ricevere commenti sul lavoro svolto.

## 3 Implementazione delle tabelle

### 3.1 Analisi della struttura già esistente delle tabelle

Terminata la fase di formazione sulle tecnologie front-end utilizzate, è stata effettuata la configurazione in locale del progetto sui cui lavorare seguendo questi passi:

1. Creare un account su Bitbucket.
2. Generare una chiave SSH da utilizzare in Bitbucket, procedura utile per evitare di dover fornire la password ogni volta che viene fatta una push del codice. Per fare ciò è necessario eseguire da terminale i seguenti comandi nella home directory:

(a) *ssh-keygen*

(b) *ssh-add ~/.ssh/id\_rsa*

Dopo aver eseguito questi due comandi basta copiare la chiave pubblica (*~/.ssh/id\_rsa.pub*) all'interno delle impostazioni personali dell'account di Bitbucket.

3. Clonare il progetto all'interno della macchina virtuale in locale
  - (a) *git clone git@bitbucket.org:SYS\_AIT/elearning-responsive-table.git*
4. Creare su Bitbucket un nuovo branch, denominato *feature responsive*, per preservare il progetto originale in caso di problemi.
5. Spostarsi nella directory che contiene il progetto

(a) *cd learning-responsive-table*

6. Impostare come branch su cui operare quello creato al punto 4

(a) *git checkout feature-responsive*

7. Lanciare l'applicazione

(a) *make dev*

Questo comando manda in esecuzione l'applicazione sulla porta 8999, aprendola nel browser di default del proprio computer; per lo sviluppo di questo progetto si è utilizzato Chrome.

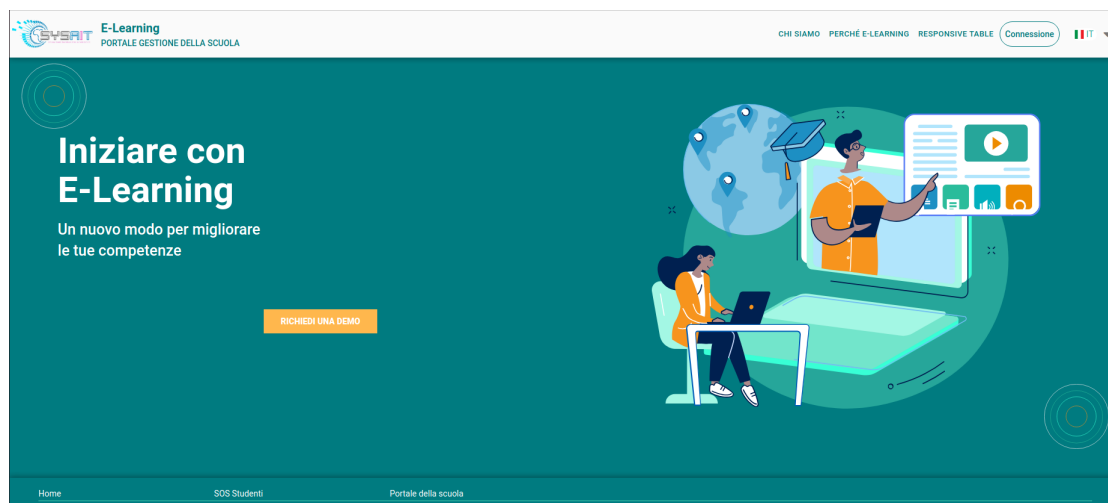


Figura 3.1.1 Schermata principale dell'applicazione

Il lavoro si è concentrato su una versione parziale del progetto, dalla quale era possibile accedere solamente ad una tabella specifica della piattaforma. Una volta terminato il progetto le modifiche effettuate sono state poi applicate alla versione più aggiornata dell'applicazione. La tabella in questione viene aperta cliccando sulla voce del menù di navigazione "responsive table" visibile nella figura 3.1.1.

<div> <div>Q Search</div> <div> <div></div> <div></div> <div></div> </div> </div>							
fullname	birthdate	Address	parent	fees_status	enrollment_date	enrollment_status	repeating
<div> <div></div> <div>MARIE NOEL ASONNA NDIE</div> <div>JAPQMGMMXABFNXY</div> <div>19610970</div> <div>TERMINAL 3</div> <div>+39 3271852672/+49 15206657219</div> <div>firstname1961.lastname1379@gmail.com</div> <div>F</div> </div>	14/09/2023 Douala (Cameroon)	Quartier 5 Tonga (Cameroon)	<div> <div>MARIE NOEL ASONNA NDIE</div> <div>+39 3271852672/+49 15206657219</div> <div>firstname1961.lastname e1379@gmail.com</div> <div>Quartier 5 Tonga (Cameroon)</div> </div>	paid	29/09/2023	inactive	
<div> <div></div> <div>PHILIPPE ATANGANA</div> <div>9EZWVRPDRFCZQZK</div> <div>24597205</div> <div>first classroom</div> <div>+39 3271852672/+49 15206657219</div> <div>firstname9666.lastname2122@gmail.com</div> <div>F</div> </div>	14/09/2023 Douala (Cameroon)	Quartier 5 Tonga (Cameroon)	<div> <div>PHILIPPE ATANGANA</div> <div>+39 3271852672/+49 15206657219</div> <div>firstname9666.lastname e2122@gmail.com</div> <div>Quartier 5 Tonga (Cameroon)</div> </div>	unpaid	29/09/2023	pending	

Figura 3.1.2

La tabella in figura 3.1.2 rappresenta i dati degli studenti iscritti in una scuola in uno specifico periodo scolastico. Al suo interno sono contenute le informazioni relative a:

- Dati anagrafici e contatti dello studente
- Data e luogo di nascita
- Indirizzo di residenza
- Dati anagrafici e contatti del genitore
- Stato del pagamento delle tasse
- Data di iscrizione
- Stato iscrizione
- Repeating (dato che avvisa se lo studente sta ripetendo l'anno)

È importante precisare che si è lavorato sul progetto senza il back-end, quindi i dati visibili a schermo nella figura 3.1.2 non provengono dal database ma sono preimpostati. Infatti non è stato possibile provare direttamente le funzioni di filtraggio, ricerca e ordinamento dei dati già implementate, ma solamente verificare se la chiamata verso il back-end partisse correttamente senza però poterne vedere il risultato.

Nonostante Quasar offra dei componenti specifici per la creazione di tabelle, come ad esempio q-table o q-card, in E-learning si è scelto un approccio diverso. In pratica sono utilizzati una serie di Single File Components che permettono di gestire ogni singolo aspetto della pagina mostrata in figura 3.1.2, quindi non solo la tabella stessa ma anche gli altri elementi visibili a schermo.

```

1 <template>
2   <div
3     :class="`${options.name} column q-gutter-y-sm`"
4     :id="`${tableName}-table`"
5   >
6     <HsTableFilters
7       @subHeaderClick="TBL_subHeaderClick"
8       :key="table_filtersKey"
9       :argument="argument"
10      class="col"
11      v-bind="table_filters"
12      v-if="table_filters"
13    />
14    <HsTableHeader
15      :key="table_headerKey"
16      class="col"
17      v-if="$q.screen.gt.sm && table_header"
18      :header="table_header"
19      :argument="argument"
20    />
21    <HsTableBody
22      :key="table_bodyKey"
23      @clickContext="TBL_clickContext"
24      @clickCell="TBL_clickCell"
25      class="col"
26      v-bind="bodyProps"
27    />
28
29    <FormPopup
30      v-if="formProps.openForm"
31      :key="`FormPopup${formProps.openForm}`"
32      v-bind="formProps"
33      @modalClosed="formClosed"
34    />
35  </div>
36

```

Figura 3.1.3 Struttura del componente HsTable

HsTable è il componente tramite il quale sono gestite tutte le tabelle presenti in E-learning. Al suo interno sono utilizzate alcune classi CSS definite da Quasar per disporre in maniera ordinata gli elementi della pagina sfruttando i flexbox.

Queste classi forniscono un modo efficiente per disporre, allineare e distribuire lo spazio tra gli elementi all'interno di un container, anche quando esso ha dimensioni sconosciute o dinamiche a tempo di esecuzione. Per poter sfruttare il meccanismo dei flexbox in Quasar è obbligatorio che nel container sia specificato come gli elementi al suo interno debbano essere posizionati. Tipicamente ciò avviene in due modi diversi, applicando al container o una classe *"column"*, come in figura 3.1.3, o una classe *"row"*. Nel primo caso i diversi componenti non possono stare sulla stessa riga, mentre nel secondo caso ciò è possibile quando le dimensioni dello schermo lo permettono.

Nei singoli elementi all'interno del container è spesso applicata la classe *col*, la quale può essere usata per specificare quanto spazio deve occupare all'interno della pagina uno specifico elemento a seconda delle caratteristiche del dispositivo con cui l'utente accede alla piattaforma.

```
<FilterPicker @updateButtons="setTableButtons"
  class="col-md-6 col-sm-12 col-xs-12"
  :argument="search.argument"
/>
```

Figura 3.1.4 Esempio di utilizzo della classe col

In figura 3.1.4 la classe col è usata per differenziare lo spazio occupato dalla barra di ricerca a seconda delle dimensioni del dispositivo. In questo caso si specifica che essa debba occupare il 100% della larghezza disponibile nei dispositivi mobili e il 50% della larghezza disponibile quando si accede da computer.

Come si può vedere in figura 3.1.3, HsTable utilizza altri 4 componenti, i quali devono prima di tutto essere importati nella parte script del file nel seguente modo:

```
import HsTableHeader from "components/utils/tables/HsTableHeader.vue";
import HsTableBody from "components/utils/tables/HsTableBody.vue";
import HsTableFilters from "components/utils/tables/HsTableFilters.vue";
import FormPopup from "src/components/forms/FormPopup.vue";
```

Tali componenti sono poi utilizzabili correttamente nel template solo se vengono aggiunti nella sezione components del file HsTable:

```
export default {
  name: "HsTable",
  components: {
    HsTableHeader,
    HsTableBody,
    HsTableFilters,
    FormPopup }
}
```



I componenti utilizzati all'interno di HsTable hanno le seguenti funzionalità:

#### 1. HsTableFilters

Gestisce la visualizzazione a schermo e l'interazione dell'utente con la barra di ricerca e con i tre bottoni che permettono rispettivamente di aggiornare il contenuto della tabella, di aggiungere una nuova riga alla tabella e di stampare il contenuto dell'intera tabella.

La barra di ricerca è gestita tramite il componente FilterPicker, il quale invia una chiamata verso il back-end ogni qualvolta l'utente vuole filtrare il contenuto di una tabella secondo una parola chiave. È presente inoltre un controllo che rifiuta la parola inserita se essa ha una lunghezza inferiore a 3 caratteri evidenziato nella figura 3.1.5.

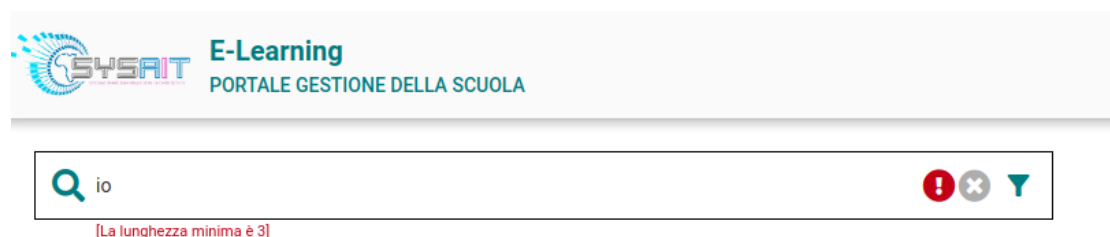


Figura 3.1.5

I bottoni descritti in precedenza e non solo sono gestiti tramite il componente ButtonComponent, il quale offre diverse proprietà con le quali è possibile personalizzare l'aspetto del bottone e il suo funzionamento. ButtonComponent sfrutta direttamente il componente per i bottoni definito da Quasar, ovvero q-btn, integrandolo con alcune proprietà personalizzate.

```
<ButtonComponent @click="() => reloadButtons(button)" textColor="primary"
  btnClass="bg-white" class="q-ml-sm" v-for="(button, index) in allButtons"
  v-bind="button" :key="`${index}buttons`">
  <q-tooltip v-if="button.tooltip" content-class="bg-primary_variant_5 text-white"
    content-style="font-size: 12px">{{ $t(button.tooltip) }}
  </q-tooltip>
</ButtonComponent>
```

Figura 3.1.6 Definizione bottoni in HsTableFilters

Come si può vedere nella figura 3.1.6 ButtonComponent viene richiamato più volte grazie alla direttiva v-for definita in Vue.js e applicata all'array allButtons nel quale sono state definite le proprietà event, icon e tooltip personalizzate per ciascun pulsante.

```
fetchButton: [
  {
    event: "fetch",
    icon: 'restart_alt',
    tooltip: "fetch"
  },
]
```

La proprietà event serve per gestire in maniera diversa ciascun bottone a seconda della sua funzionalità, mentre la proprietà icon definisce l'icona di ogni bottone sfruttando la libreria di icone Font Awesome nella sua quinta versione. Infine la proprietà tooltip viene usata per offrire maggiori informazioni all'utente quando passa il cursore su un determinato bottone. Il testo viene tradotta in maniera diversa a seconda della lingua selezionata ed è gestita grazie al componente q-tooltip definito da Quasar.

## 2. HsTableHeader

Componente dedicato alla gestione della riga di intestazione della tabella, ovvero della riga utilizzata per visualizzare i nomi dei singoli campi.

fullname	birthdate	Indirizzo	parent	fees_status	enrollment_date	enrollment_status	repeating
----------	-----------	-----------	--------	-------------	-----------------	-------------------	-----------

Figura 3.1.7 Riga della tabella gestita da HsTableHeader

Come si può vedere in figura 3.1.7 la maggior parte degli attributi non sono tradotti nonostante si sia selezionata la lingua italiana; ciò è dovuto al fatto che nei file Javascript utilizzati all'interno della piattaforma per supportare il multilinguismo non sono state ancora inserite tutte le traduzioni necessarie.

Una particolarità di questo componente è che esso non deve essere sempre inserito all'interno del DOM, ma solamente nel caso in cui l'utente stia accedendo all'applicazione da computer. Questo è necessario perché nel design previsto per dispositivi mobili che sarà descritto nel paragrafo 3.2 non è prevista la presenza della riga di intestazione. Per questo scopo si sfrutta la direttiva v-if fornita da Vue.js, la quale permette di aggiungere un certo elemento alla pagina solo al verificarsi di una o più condizioni.

Nel caso specifico, come si può vedere in figura 3.1.3, la presenza di `HsTableHeader` è determinata da due condizioni che devono essere verificate contemporaneamente, ovvero:

- a) `"$q.screen.gt.sm"`: serve per applicare quanto detto in precedenza, in pratica viene fatto un controllo sulla dimensione dello schermo e si stabilisce che `HsTableHeader` dovrà essere visualizzato solo se la larghezza dello schermo è superiore ai 1024 pixel. Questa condizione è molto importante e verrà utilizzata spesso nel paragrafo 3.2 per evitare che le modifiche fatte per rendere l'interfaccia responsive vadano a generare errori nel layout che si sta descrivendo.
- b) `"table_header"`: controllo fatto per evitare che `HsTableHeader` venga aggiunto al DOM se per qualche motivo il back-end non è stato in grado di fornire le informazioni necessarie per visualizzare correttamente il componente. Lo stesso controllo ma su diversi attributi è utilizzato anche in tutti gli altri componenti di `HsTable` per il medesimo motivo.

Un altro aspetto importante gestito all'interno del componente `HsTableHeader` è l'ordinamento dei dati. Nel caso specifico come si può vedere nella figura 3.1.7 è prevista la possibilità di ordinare gli elementi solamente rispetto al nome dello studente, ovvero rispetto al campo principale della tabella.

```
<HsTableOrder
  v-if="header.principal.sort_param"
  @click="
    (params) =>
      orderClick({ ...params, sort: header.principal.sort_param })
  "
  class="cell-icons"
  :id="`${argument}-order-0`"
/>
```

Figura 3.1.8 Gestione dell'ordinamento in `HsTableHeader`

La gestione dell'ordinamento viene in parte delegata al componente `HsTableOrder`, il quale ha un template caratterizzato da due bottoni, uno per ordinare i dati in modo crescente e l'altro per ordinare i dati in modo decrescente. Dalla figura 3.1.8 si nota che questo componente viene incluso nel DOM solamente se nel back-end è stato impostato a `true` l'attributo booleano `"sort_param"`. Questo controllo viene effettuato per tutti i campi della tabella, dando quindi la possibilità di ordinare i dati in base anche ai campi secondari se il back-end lo permette.

```
filtering(params){

  this.currentBtn = this.filterBtns.find( btn => btn.event == params.event)
  let otherBtn = this.filterBtns.find( btn => btn.event != params.event)

  if(otherBtn.activated) otherBtn.activated = false
  this.currentBtn.activated = !this.currentBtn.activated

  this.$emit("click", { order: this.currentBtn.event == "up" ?
    'asc' : 'desc', value: this.currentBtn.activated } )

  this.$hs.methods.COMMON_emitEvent(this.customEvent, this.id);
}
```

La funzione *filtering* definita in `HsTableOrder` viene invocata ogni volta che si clicca su uno dei due bottoni. Essa controlla quale dei due pulsanti è stato premuto ed evita che siano entrambi attivi nello stesso momento, poi configura due parametri da inviare ad `HsTableHeader`; il primo specifica il tipo di ordinamento, mentre il secondo è un attributo booleano che specifica se l'evento di click ha attivato o disattivato il bottone in questione.

```
orderClick(params) {

  this.form.sort = params.value
  ? this.changeArgumentUrlParams({ ...this.form,
    urlParams: { sort: `${params.sort}_${params.order}` }})
  : this.deleteFromUrlParams({ ...this.form, param: "sort" });

  this.$hs.methods.COMMON_emitEvent(`Filter${this.argument}ByParams`);
},
```

All'interno di `HsTableHeader` è presente la funzione *orderClick*, la quale controlla se il bottone è stato attivato o disattivato; nel primo caso vengono aggiunti alla chiamata verso il back-end i parametri necessari per l'ordinamento, ovvero il tipo di ordinamento e il campo su cui è applicato. Nel secondo caso invece i parametri in questione vengono rimossi.

### 3. HsTableBody

Componente che si occupa della visualizzazione di tutte le righe della tabella, è contraddistinto da una struttura gerarchica ben precisa che permette di gestire ogni cella singolarmente.

```
<template v-if="thereAreElements">
  <HsTableRow
    v-for="(element, index) in allElements"
    @clickCell="TBL_clickCell"
    :rowElement="element"
    :key="index"
    class="col-md-12 col-sm-12 col-xs-12"
  />
</template>
```

Figura 3.1.9 Struttura di HsTableBody

Per prima cosa si verifica che la tabella richiesta non sia vuota, se questa condizione è vera viene utilizzato più volte il componente HsTableRow sfruttando la direttiva v-for come si vede in figura 3.1.9. Tale direttiva viene applicata sull'array allElements proveniente dal back-end e contenente tutti gli elementi della tabella.

Tramite le classi CSS descritte in precedenza viene specificato inoltre che su qualunque dispositivo questo componente debba occupare in larghezza tutto lo spazio disponibile.

```
TBL_clickCell(response){
  const { event, data } = response;
  switch (event) {
    case 'contextMenu':
      this.contextMenu_fn(response, data, this.id);
      break;
    default:
      this.$emit("clickCell", response);
      break;
  }
},
```

L'evento clickCell, associato al metodo TBL\_clickCell, gestisce l'interazione con la singola riga della tabella permettendo ad esempio di accedere alla pagina del singolo studente cliccando su nome e cognome. Nella versione utilizzata per la realizzazione del progetto la pagina viene aperta correttamente ma risulta vuota a causa dell'assenza del back-end.

```

<template>
  <div :class="`${$options.name}`" v-if="rowElement">
    <component @clickCell="TBL_clickCell"
      :is="$q.screen.gt.sm ? 'HsTableRowLine' : 'HsTableRowCard' " :element="rowElement">
    </component>
  </div>
</template>

```

Figura 3.1.11 Struttura di HsTableRow

Come si può vedere in figura 3.1.11 all'interno di HsTableRow viene utilizzata la stessa condizione vista in HsTableHeader per essere in grado di utilizzare un componente diverso a seconda del tipo di dispositivo utilizzato. Quando si accede all'applicazione da computer viene invocato il componente HsTableRowLine, mentre quando si accede da mobile viene invocato il componente HsTableRowCard. La maggior parte delle modifiche per rendere l'interfaccia responsive descritte nel paragrafo 3.2 riguarderanno proprio l'implementazione di quest'ultimo componente.

#### 4. FormPopup

Componente utilizzato nel caso in cui l'utente voglia aggiungere un nuovo elemento alla tabella oppure modificarne uno esistente.

In pratica a seconda dell'evento generato e della tabella su cui si sta operando FormPopup apre all'interno della stessa pagina un form con diverse caratteristiche e campi da compilare. Questo è possibile solo tramite il supporto del back-end, pertanto non è possibile mostrarne il corretto funzionamento.

In ogni caso non sono state considerate necessarie modifiche a questo componente ai fini del progetto al centro di questa tesi.

### 3.2 Modifiche per rendere l'interfaccia responsive

L'implementazione definita nel paragrafo precedente prevede che ogni tabella della piattaforma possa contenere un elevato numero di dati, una situazione quindi difficile da adattare su un dispositivo di piccole dimensioni. Per questo motivo risulta necessario utilizzare un design differente per dispositivi mobili, in modo tale da migliorare l'esperienza utente e la leggibilità dei dati.

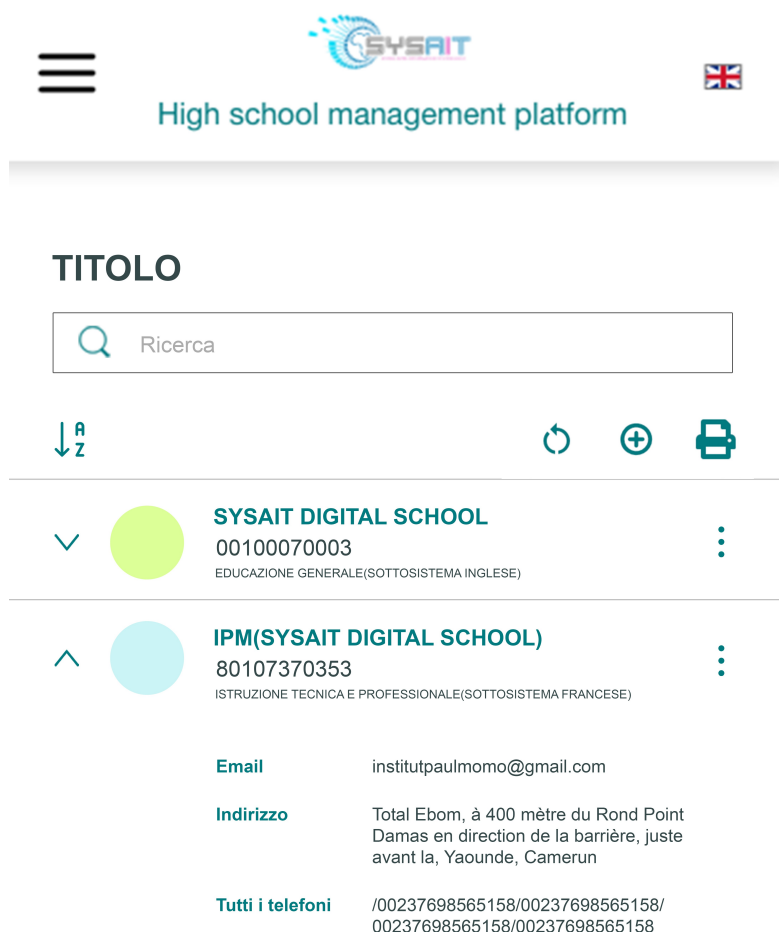


Figura 3.2.1

In figura 3.2.1 si può vedere il design proposto per i dispositivi mobili, in questo caso applicato ad una tabella contenente le scuole iscritte alla piattaforma. Tale layout prevede che il comportamento di default, per ogni riga della tabella, sia quello di mostrare a schermo solamente i dati principali del singolo elemento; successivamente l'utente ha la possibilità di accedere ai dati secondari tramite un bottone. Tali dati devono rigorosamente essere visualizzati sotto ai dati principali senza posizionarsi sotto all'immagine o agli altri pulsanti.

Un altro aspetto già evidenziato in precedenza è che da mobile non deve essere visibile la riga di intestazione della tabella gestita dal componente `HsTableHeader`, questo però va ad eliminare anche la parte della tabella dedicata alla gestione dell'ordinamento dei dati. Sarà quindi necessario implementare tale funzionalità nel componente `HsTableFilters` e gestirla in maniera leggermente diversa tramite l'utilizzo di bottoni messi sulla stessa riga di quelli presenti anche nel design descritto al paragrafo 3.1.

Prima di descrivere le modifiche utilizzate per realizzare il design proposto vengono riportati i comandi utilizzati più di frequente da terminale per salvare periodicamente le modifiche apportate al progetto su Bitbucket.

- *"git status"*: comando utilizzato per verificare lo stato della directory del progetto, serve per visualizzare quali modifiche sono state effettuate rispetto all'ultimo salvataggio effettuato sul branch di riferimento.
- *"git add ."*: primo comando da eseguire quando si vuole effettuare un commit del progetto. È usato per aggiungere tutte le modifiche effettuate alla cosiddetta staging area, ovvero il luogo in cui vengono raccolte tutte le modifiche che faranno parte del prossimo commit. Se si desidera aggiungere solo alcuni file al commit in questione è necessario specificarne il nome dopo `git add`.
- *"git commit -m " messaggio di commit" "*: comando utilizzato per aggiungere al commit del progetto una breve descrizione delle modifiche effettuate.
- *"git push"*: comando con il quale viene completata la procedura di aggiornamento del branch caricando su Bitbucket la versione aggiornata del progetto con relativo commento.
- *"git status"*: comando utilizzato per verificare che il progetto in locale sia aggiornato all'ultimo commit effettuato su Bitbucket.
- *"git pull"*: comando utilizzato per aggiornare il progetto in locale se sono state rilevate modifiche su Bitbucket.
- *"git reset --soft HEAD~1"*: comando utilizzato in un paio di occasioni prima di `git push` per annullare l'ultimo commit ed evitare di aggiungere su Bitbucket alcune modifiche non compatibili con la versione più aggiornata del progetto.



Il primo passo del lavoro è stato quello di implementare la funzione di ordinamento per dispositivi mobili; per fare ciò è stato necessario creare un nuovo componente chiamato HsTableOrderMobile e utilizzato all'interno di HsTableFilters.

```
<div v-if="!$q.screen.gt.sm" class="left-buttons">
  <HsTableOrderMobile v-if="header.principal.sort_param"
    @click="(params) =>
      | orderClick({ ...params, sort: header.principal.sort_param })
      | "
    :id="`${argument}-order-0`"/>
</div>
```

Figura 3.2.2

Come si può vedere in figura 3.2.2 la presenza di HsTableOrderMobile dipende dalla dimensione dello schermo del dispositivo utilizzato; viene aggiunta inoltre una classe personalizzata CSS denominata "left-buttons" per evitare che i bottoni dedicati alla gestione dell'ordinamento siano posizionati vicino ai bottoni già presenti.

Si precisa che da dispositivo mobile sarà possibile effettuare l'ordinamento solamente in base al dato primario di ogni tabella (nel caso specifico il nome dello studente).

```
data () {
  return {
    sortButtonUp: {
      tooltip: "orderUp",
      event: "up",
      icon: 'fas fa-sort-alpha-up',
      actived: false,
      isVisible: false,
    },
    sortButtonDown: {
      tooltip: "orderDown",
      event: "down",
      icon: 'fas fa-sort-alpha-down',
      actived: false,
      isVisible: true
    },
    cancelOrderButton:
    {
      tooltip: "cancelOrder",
      icon: 'far fa-times-circle',
      event: 'cancel'
    },
  },
};
```

Nella parte di script di `HsTableOrderMobile` vengono inseriti i parametri dei bottoni necessari per gestire l'ordinamento. Questi elementi vengono inseriti all'interno della sezione `data` dello script, ovvero la memoria locale di ogni componente dove possono essere salvate tutte le variabili necessarie per il suo corretto funzionamento.

Questi dati sono poi associati a tre diversi `ButtonComponent` definiti all'interno del template. Sono definiti due bottoni distinti per l'ordinamento crescente e per l'ordinamento decrescente, inizialmente entrambi non sono attivi e viene mostrato a schermo solamente il pulsante per l'ordinamento decrescente, ovvero `sortButtonDown`. I due pulsanti per ordinare i dati non devono mai essere visualizzati a schermo contemporaneamente, inoltre il bottone `cancelOrderButton` deve essere visibile solo se uno dei due bottoni di ordinamento risulta attivo.

In questo caso si è scelto di gestire la visualizzazione a schermo di ciascun bottone tramite la direttiva `v-show` definita da `Vue-Js`; essa ha una funzionalità simile alla direttiva `v-if` ma è adatta in casi come questo dove il risultato della condizione può cambiare spesso a tempo di esecuzione. Utilizzando `v-show` infatti i componenti vengono renderizzati e inseriti nel DOM in ogni caso, quando il risultato della condizione cambia viene solamente modificata la proprietà CSS di `display` dell'elemento in questione.

```
<ButtonComponent v-show="sortButtonDown.active || sortButtonUp.active"
  v-bind="cancelOrderButton" textColor="primary" btnClass="bg-white" class="q-ml-sm"
  @click="(params) => filtering({ params, event: cancelOrderButton.event })">
  <q-tooltip v-if="cancelOrderButton.tooltip" content-class="bg-primary_variant_5 text-white"
    content-style="font-size: 12px">{{ $t(cancelOrderButton.tooltip) }}</q-tooltip>
</ButtonComponent>
```

Figura 3.2.3 Definizione del bottone `cancelOrderButton` nel template

Come si può vedere in figura 3.2.3 gli attributi definiti in precedenza vengono poi associati al rispettivo `ButtonComponent` tramite la direttiva `v-bind`. Per ogni bottone sono definite delle proprietà per personalizzarne il colore e una classe definita da `Quasar` per aggiungere un piccolo margine a sinistra.

Quest'ultima classe fa parte delle cosiddette `CSS Spacing Classes`, utilizzate per aggiungere margine ad un elemento del template potendo personalizzare la direzione e la dimensione del margine stesso.

L'interazione dell'utente con ogni bottone è gestita anche in questo caso tramite la funzione `filtering`, che adotta tuttavia un'implementazione leggermente diversa rispetto a quella vista in precedenza.

```
if(params.event=="cancel"){
    this.sortButtonDown.activated = false
    this.sortButtonDown.isVisible = true
    this.sortButtonUp.activated = false
    this.sortButtonUp.isVisible = false
    this.$emit("click", { order: this.currentBtn.event == "up" ?
        'asc' : 'desc', value: false })
}
```

Al codice della funzione `filtering` viene aggiunta la gestione dell'evento `cancel` generato quando viene premuto il pulsante `cancelOrderButton`, il quale ripristina i valori iniziali degli attributi dei due bottoni di ordinamento. Dopo aver fatto questo vengono configurati i parametri da inviare a `HsTableFilters` per processare correttamente la richiesta dell'utente.

```
if (params.event == "up") {
    this.currentBtn = this.sortButtonUp
    var otherBtn = this.sortButtonDown

}
if (params.event == "down") {
    this.currentBtn = this.sortButtonDown
    var otherBtn = this.sortButtonUp
}

otherBtn.isVisible = true
this.currentBtn.isVisible = false

otherBtn.activated = false
this.currentBtn.activated = true

this.$emit("click", { order: this.currentBtn.event == "up" ?
    'asc' : 'desc', value: this.currentBtn.activated })
```

Quando viene premuto il pulsante di ordinamento invece la gestione è simile a quanto visto in precedenza, aggiungendo solamente la gestione dell'attributo `isVisible`. In pratica si imposta che il bottone appena premuto non sarà più visibile, lasciando spazio al bottone per effettuare l'ordinamento in maniera opposta rispetto a quanto appena fatto. Si evita quindi che entrambi i bottoni siano visibili contemporaneamente.

In `HsTableFilters` invece viene aggiunta la funzione `orderClick` con la stessa implementazione vista al paragrafo 3.1.

L'ultimo aspetto da gestire per ogni bottone è stato l'attributo `tooltip`, per il quale sono state aggiunte le traduzioni in tutte e tre le lingue supportate da E-learning ai rispettivi file Javascript come riportato nelle figure 3.2.4, 3.2.5 e 3.2.6.

```
orderDown: 'Descending order',  
orderUp: 'Ascending order',  
cancelOrder: 'Remove order',
```

Figura 3.2.4 Traduzioni in inglese

```
orderDown: 'Ordre descendant',  
orderUp: 'Ordre ascendant',  
cancelOrder: 'Annulle ordre',
```

Figura 3.2.5 Traduzioni in francese

```
orderDown: 'Ordinamento decrescente',  
orderUp: 'Ordinamento crescente',  
cancelOrder: 'Cancella ordinamento',
```

Figura 3.2.6 Traduzioni in italiano

La situazione quando si clicca sul pulsante di ordinamento per la prima volta per ordinare i dati in maniera decrescente è quindi la seguente:

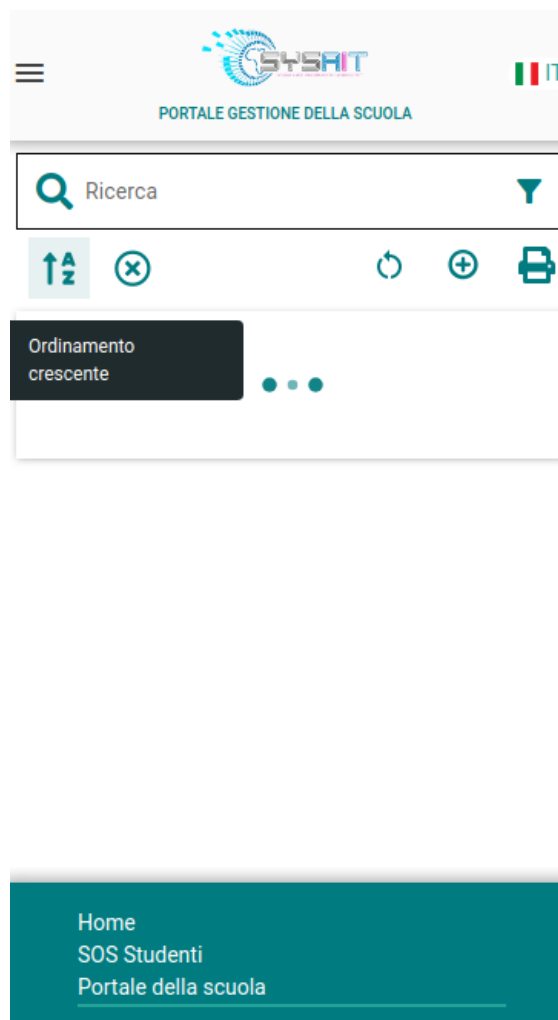


Figura 3.2.7

Come si vede in figura 3.2.7 la chiamata verso il back-end parte senza errori, lasciando spazio al pulsante per eseguire l'ordinamento in senso opposto ed al pulsante per cancellare il criterio applicato.

Come già detto in precedenza a causa dell'assenza del back-end non è stato possibile riportare i risultati dell'ordinamento, ciò nonostante l'implementazione appena descritta è stata approvata.

Il passaggio successivo consiste nell'implementazione del componente `HsTableRowCard`, necessario per gestire la visualizzazione dei diversi elementi della tabella.

```
<div class="row full-width items-start no-wrap">
  <div class="col-1 hsl-caret q-mr-xs row justify-center">
    <q-btn @click="isClicked = !isClicked" flat round color="primary"
      dense size="sm" style="font-size: 12px" :icon="`fas fa-chevron-${isClicked ? 'up' : 'down'}`" />
    </div>
    <HsTableImage :class="`image col-2 ${element.image.dimenssion} || ''`" v-if="element.image" :image="element.image"/>
    <HsTablePrincipal @clickCell="TBL_clickCell" style="word-wrap: anywhere"
      :class="`col-8 principal table-principal-mobile ${element.principal.dimenssion} || ''`"
      v-if="element.principal" :principal="element.principal"/>
    <HsTableButtons @click="(evt) => TBL_clickCell({ evt, event: element.buttons.event, data: element.buttons.data })"
      :class="`col-1 items-end ${element.buttons.dimenssion} || ''`" v-if="element.buttons" v-bind="element.buttons"/>
  </div>
```

Figura 3.2.8 Struttura di `HsTableRowCard`

La struttura di `HsTableRowCard` evidenziata in figura 3.2.8 permette di suddividere ciascuna riga della tabella in quattro parti, sfruttando alcuni componenti utilizzati anche nell'implementazione per computer; usufruendo del meccanismo dei flexbox si impone che questi componenti debbano stare sulla stessa riga tramite la classe `row`, assegnando poi a ciascun elemento una dimensione massima occupabile all'interno della riga.

Il risultato ottenuto tramite questa implementazione è il seguente:



Figura 3.2.9

Tutti i dati rappresentati in figura 3.2.9 sono accessibili tramite l'oggetto `element`, il quale viene ereditato dal componente `HsTableRow` che a sua volta l'ha ottenuto da `HsTableBody`. Tale oggetto è definito all'interno della parte script del Single File Component nel seguente modo:

```
props: {
  element: { type: Object, required: true },
},
```

L'oggetto `element` viene definito nella sezione `props` dello script, la quale è dedicata ai parametri del componente figlio che sono stati passati dal componente padre.

I quattro componenti utilizzati in `HsTableRowCard` hanno le seguenti funzionalità:

1. Il primo componente utilizzato è un bottone per visualizzare i dati secondari gestito tramite il componente `q-btn`; in questo caso si è scelto di non utilizzare `ButtonComponent` per gestire in maniera più semplice l'evento di click del mouse. Quando ciò si verifica viene cambiato il valore dell'attributo booleano `isClicked` definito nel seguente modo:

```
data () {  
  return {  
    isClicked: false  
  }  
},
```

Questo dato, come si vedrà in seguito, è direttamente associato alla parte del componente dedicata alla gestione dei dati secondari di ciascun elemento. Dal valore di questo attributo dipende anche l'icona che viene mostrata a schermo.

Al bottone sono poi associate una serie di classi CSS tra cui la classe `col-1`, la quale fornisce al componente un dodicesimo della larghezza della riga. Viene poi sfruttata la classe `q-mr-xs` per applicare un margine a destra dell'elemento di dimensioni ridotte per distanziarlo dall'immagine.

2. Il secondo componente utilizzato è `HsTableImage`, il quale permette nel caso specifico di visualizzare a schermo l'immagine profilo associata allo studente. Si utilizza la classe `col-2` per assegnare all'immagine un sesto della larghezza della riga, occupando quindi il doppio dello spazio rispetto al bottone visto in precedenza. Non tutte le tabelle utilizzate in E-learning fanno utilizzo di questo componente, pertanto esso viene aggiunto al DOM solo quando l'oggetto `image` è presente. Se la condizione è vera esso viene passato come parametro a `HsTableImage`, il quale ha la seguente struttura:

```
<q-avatar  
  class="text-white q-mx-xs"  
  :size="avatarSize"  
  
  :class="{ 'cursor-pointer': image.event, 'bg-primary': !image.url }"  
  @click="(evt) => {{ if (image.event) $emit('click', { evt, event: image.event, data: image.data }); }}"  
>  
    
  <span v-else class="text-bold">{{ image.fallback }}</span>  
</q-avatar>
```

Figura 3.2.10 Struttura di `HsTableImage`

Il componente `q-avatar` utilizzato in figura 3.2.10 permette di creare un elemento scalabile che può contenere al suo interno testo, icone o immagini. L'impostazione di default utilizzata in questo caso fornisce all'elemento una forma circolare.

Se l'oggetto image possiede l'attributo url all'interno del componente q-avatar viene inserita l'immagine associata, mentre se l'attributo url non è presente viene inserito un testo di fallback all'interno del componente.



Figura 3.2.11 Esempio di fallback

Come si può vedere nella figura 3.2.11 il fallback in questo caso prevede, quando l'url dell'immagine non è disponibile, di inserire al suo posto le lettere iniziali del nome e del cognome dello studente.

3. Il terzo componente utilizzato è HsTablePrincipal, il quale è dedicato alla gestione dei dati principali di ciascun elemento della tabella. Ad esso è applicato lo stile *word-wrap: anywhere* per forzare il testo ad andare a capo in qualunque momento, evitando quindi che parole lunghe possano portare il componente a superare le dimensioni massime che gli sono state assegnate; tramite la classe *col-8* tali dimensioni vengono impostate a due terzi della larghezza della riga.

```
<template>
  <div :class="`${$options.name}`">
    <HsTableCell @clickCell="TBL_clickCell" :value="principal"/>
  </div>
</template>
```

Figura 3.2.12 Struttura di HsTablePrincipal



In figura 3.2.12 si può vedere che `HsTablePrincipal` delega la gestione dei dati principali al componente `HsTableCell`, il quale viene sfruttato anche per la visualizzazione dei dati secondari. Non è possibile utilizzare direttamente quest'ultimo componente nel template di `HsTableRowCard` perchè all'interno di `HsTablePrincipal` vengono applicati degli stili CSS personalizzati ai dati principali, definiti nella parte style del file.

```
<style lang="scss">
  .HsTablePrincipal {
    .HsTableCell {
      font-family: 'Philosopher', sans-serif;
      font-size: 0.7rem;
    }
    .HsTableMicroCell:first-child {
      font-size: 0.8rem;
      text-transform: uppercase;
    }
  }
}</style>
```

Oltre ad impostare la dimensione e il tipo di font da applicare, questa classe CSS stabilisce che la dimensione del carattere del primo dato, in figura 3.2.11 il nome dello studente, sia maggiore rispetto agli altri.

Come accennato in precedenza `HsTableCell` viene utilizzato nel template di `HsTableRowCard` per gestire anche la visualizzazione dei dati secondari:

```
<div v-show="isClicked">
  <HsTableCell
    @clickCell="TBL_clickCell"
    :class="`table-cell-mobile`"
    v-for="(detail, index) in element.details"
    v-bind="detail"
    :key="index"
  />
</div>
```




Figura 3.2.13

Come si può vedere in figura 3.2.13 la presenza di questo componente è legata al valore dell'attributo `isClicked` gestito dal bottone descritto in precedenza.



```
.table-cell-mobile{
  font-size: 12px;
  max-width: 100%;
}
```

All'elemento è inoltre applicata la classe CSS personalizzata `table-cell-mobile` per impostare la dimensione del carattere e stabilire che l'elemento dovrà sempre occupare la massima larghezza disponibile.


Si può notare anche che, a differenza di quanto avviene con `HsTablePrincipal`, il componente `HsTableCell` viene chiamato grazie alla direttiva `v-for` tante volte quanti sono i dati secondari della tabella (nel caso specifico il componente viene richiamato sette volte). Il motivo per cui la chiamata avviene in modi diversi è che i dati principali sono memorizzati all'interno di un singolo oggetto, mentre i dati secondari sono memorizzati all'interno di una lista di oggetti.



 EN

HIGH SCHOOL MANAGEMENT PLATFORM

**ANDRE TIENTCHEU ZOA**  
 WJBK0XWDMXWABU  
**68097710**  
**first classroom**  
 +39 3271852672/+49 15206657219  
 firstname1870.lastname2800@gmail.com  
 M



**birthdate** 14/09/2023 Douala  
(Cameroon)

**Address** Quartier 5 Tonga  
(Cameroon)

**parent** ANDRE TIENTCHEU ZOA  
 +39 3271852672/+49  
 15206657219  
 firstname1870.lastname2  
 800@gmail.com  
 Quartier 5 Tonga  
(Cameroon)

**fees\_status** ● unpaid

**enrollment\_** 29/09/2023  
**date**

**enrollment\_** ● pending  
**status**

**repeating** ●

Figura 3.2.14

In figura 3.2.14 si nota un'altra importante differenza tra dati principali e dati secondari, ovvero che a questi ultimi è associato anche un attributo label; esso viene sfruttato all'interno di HsTableCell per trattare dati secondari e principali in modo leggermente diverso.

```

<template>
  <div v-bind="calculateAttributes" v-if="value">
    <div :class="{ 'row justify-end full-width table-cell-mobile': !$q.screen.gt.sm && label}">
      <span v-if="!$q.screen.gt.sm && label" class="label-card col-4" >{{t(label)}}</span>
      <HsTableMicroCell @clickCell="TBL_clickCell"
        class="col-8"
        v-for="(element, index) in value.list"
        v-bind="element"
        :key="index"
        :label="label" />
    </div>
  </div>
</template>

```

Figura 3.2.15 Struttura di HsTableCell

All'interno di HsTableCell viene ancora una volta utilizzato il meccanismo dei flexbox per i dati secondari, ma solo quando la dimensione dello schermo è inferiore ai 1024 pixel ed è definito l'attributo label.

Queste due condizioni sono fondamentali per evitare che queste modifiche si riflettano sui dati principali e sull'implementazione delle tabelle per computer. Quando questi due requisiti sono verificati si impone quindi che lo spazio riservato al label sia di un terzo rispetto alla larghezza disponibile e che i dati secondari occupino i due terzi rimanenti.

```

.table-cell-mobile{
  word-wrap: break-word;
  margin-left: calc(25% + 5px);
  margin-right: 8.333%;
}

```

Questa classe serve per imporre dei margini che mantengano i dati secondari sempre sotto i dati primari, senza andare mai a invadere lo spazio sotto all'immagine o ai bottoni. Determinare i margini in questo modo è possibile solo grazie al flexbox applicato in HsTableRowCard che definisce delle dimensioni fisse per ciascun elemento della tabella.

Il componente HsTableMicroCell utilizzato in figura 3.2.15 è particolarmente utile per entrambi i tipi di dato:

- Nel caso dei dati primari la direttiva v-for applicata a questo componente permette di visualizzare correttamente tutti i campi che devono essere sempre visualizzati a schermo. Senza HsTableMicroCell nel caso di questa tabella verrebbe visualizzato solamente il nome e cognome dello studente.
- Nel caso dei dati secondari invece HsTableMicroCell è necessario per quei campi che sono rappresentati da più di un elemento in cui un dato secondario sia rappresentato da più di un elemento, come ad esempio nel caso di fees\_status o enrollment\_status.

4. L'ultimo componente utilizzato è HsTableButtons, il quale gestisce il bottone di ellipsis tramite il quale è possibile accedere ad alcune operazioni specifiche sulla singola riga come si può vedere nella seguente figura:








✓		<b>MARIE NOEL ASONNA NDIE</b> JAPQMGMX4BFNXY <b>19610970</b> <b>TERMINAL 3</b> +39 3271852672/+49 15206657219 firstname1961.lastname1379@gmail.com F	<div> Print  Update  Delete  Login as <b>Marie Noel Asonna Ndie</b></div>
✓		<b>PHILIPPE ATANGANA</b> 9E2WRPD8FCZOK <b>24597205</b> <b>first classroom</b> +39 3271852672/+49 15206657219 firstname9666.lastname2122@gmail.com F	

Figura 3.2.16


Come per il primo componente a questo elemento è associata la classe *col-1*, con la quale viene quindi dato al bottone un dodicesimo della larghezza della riga.


Il risultato completo dell'implementazione appena descritta è il seguente:











PORTALE GESTIONE DELLA SCUOLA




 Ricerca









**MARIE NOEL ASONNA NDIE**  
JAPQMGMX4BFNXY  
**19610970**  
**TERMINAL 3**  
+39 3271852672/+49 15206657219  
firstname1961.lastname1379@gmail.com  
F

birthdate

14/09/2023 Douala (Cameroon)

Indirizzo

Quartier 5 Tonga (Cameroon)

parent

MARIE NOEL ASONNA NDIE  
+39 3271852672/+49 15206657219  
firstname1961.lastname1379@gmail.com  
Quartier 5 Tonga (Cameroon)

fees\_status

● paid

enrollment\_date


29/09/2023


enrollment\_status

● inactive

repeating

●





**PHILIPPE ATANGANA**  
9E2WRPD8FCZOZK  
**24597205**  
**first classroom**  
+39 3271852672/+49 15206657219  
firstname9666.lastname2122@gmail.com  
F

Figura 3.2.17

38

Vengono riportate infine alcune immagini di un'altra tabella ottenute una volta che l'implementazione è stata aggiunta al progetto più aggiornato di E-learning:

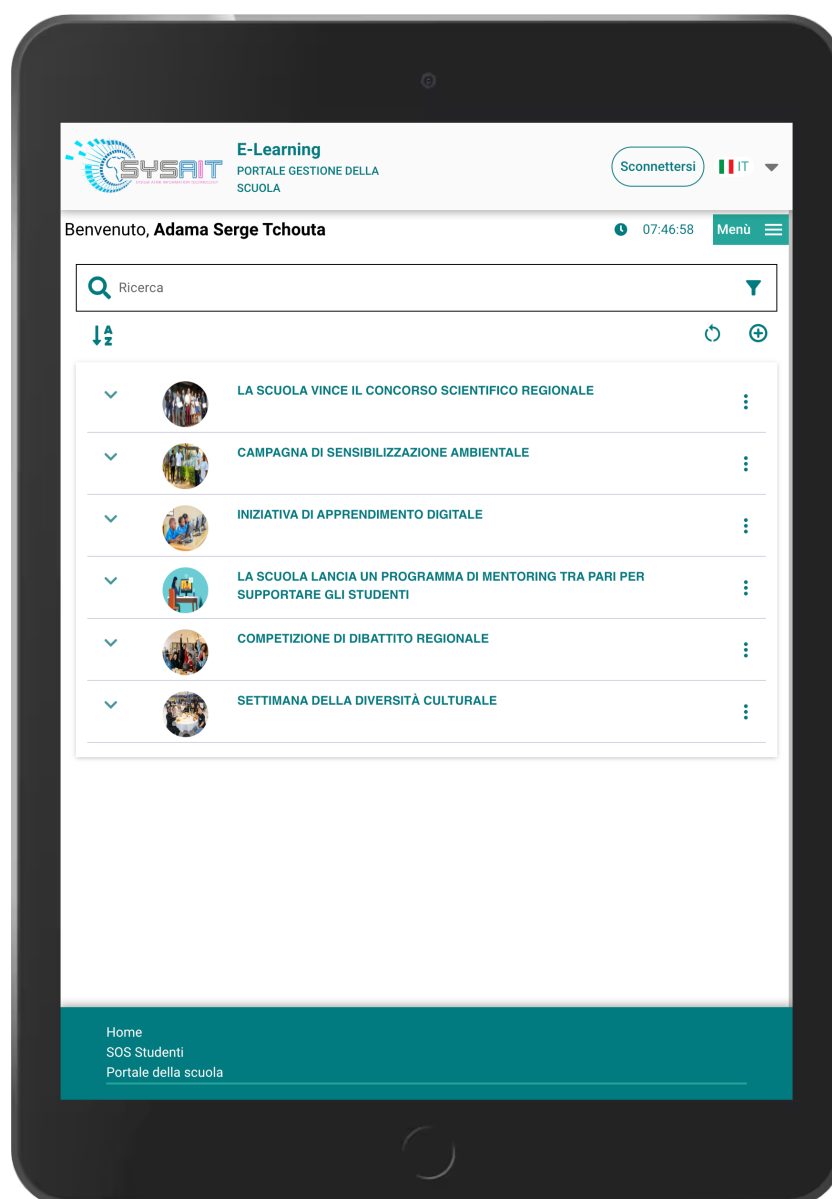


Figura 3.2.18

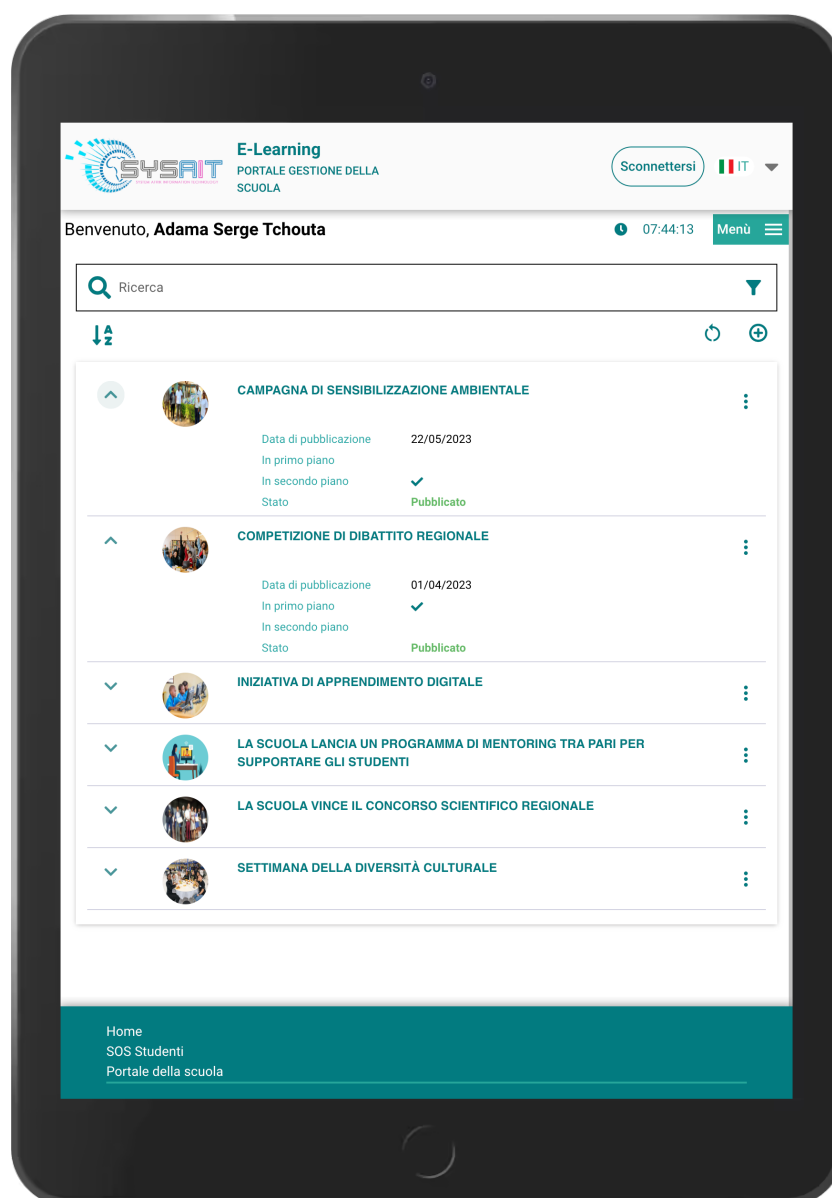


Figura 3.2.19

Le figure 3.2.18 e 3.2.19 mostrano come appare la tabella su un dispositivo con dimensioni dello schermo paragonabil ad un iPad.





## Conclusione

Questo progetto rappresenta solo un piccolo contributo ad una piattaforma così complessa e ambiziosa come quella in sviluppo da SYSAIT.

Il lavoro descritto in queste pagine ha permesso di raggiungere praticamente tutti gli obiettivi di progetto prefissati, ottenendo quindi un'interfaccia per le tabelle della piattaforma E-learning che sia facilmente leggibile ed utilizzabile da qualunque dispositivo.

Il progetto è stato sicuramente in parte facilitato da una struttura per le tabelle ben definita che ha permesso di arrivare ai risultati sperati senza dover modificare eccessivamente l'architettura già presente.

L'unico aspetto che non è stato possibile approfondire riguarda il comportamento da mobile di alcune operazioni effettuabili all'interno della pagina dedicata alla tabella, le quali avrebbero richiesto la presenza del back-end per poter essere valutate completamente.



## Sitografia

1. <https://www.sysaitechnology.com/>
2. <https://v2.vuejs.org/v2/guide/index.html-What-is-Vue-js>
3. <https://v2.vuejs.org/v2/guide/conditional.html>
4. <https://v1.quasar.dev/quasar-cli/developing-spa/introduction>
5. <https://v1.quasar.dev/start/how-to-use-vue>
6. <https://v1.quasar.dev/introduction-to-quasar>
7. <https://v1.quasar.dev/vue-components/button>
8. <https://v1.quasar.dev/vue-components/tooltip>
9. <https://v1.quasar.dev/layout/grid/introduction-to-flexbox>
10. <https://v1.quasar.dev/layout/grid/row>
11. <https://v1.quasar.dev/style/spacing>
12. <https://v1.quasar.dev/vue-components/avatar>
13. <https://www.postgresql.org/about/>



## Ringraziamenti

Mi prendo quest'ultima pagina per citare le persone che mi hanno accompagnato in questo percorso universitario, rendendolo più semplice e piacevole.

Prima di tutto ci tengo a ringraziare la mia famiglia per essermi stato vicino in questi tre anni e mezzo, senza mai mettermi troppa pressione e supportandomi sempre. Senza di loro raggiungere questo importante traguardo non sarebbe stato possibile.

Un ringraziamento speciale agli amici di una vita, in particolar modo a quelli che hanno condiviso prima o dopo il percorso universitario con me. Li ringrazio soprattutto per la loro presenza nel primo anno di questo percorso, condizionato dalla pandemia, fondamentale per proseguire negli studi in un periodo così complesso. Ci tengo a citare in particolar modo Federico, con il quale si è creato un rapporto di amicizia che dura da una vita ormai e che ci sta portando a concludere questo percorso insieme.

Desidero ringraziare anche tutte le persone conosciute grazie all'università con le quali ho condiviso l'ansia per gli esami e che mi hanno sicuramente aiutato in un momento della vita fondamentale.

Un ringraziamento particolare va fatto anche all'ingegnere Wilfried Ndomo Kenfack per la costante presenza sia durante la fase di apprendimento delle conoscenze necessarie per il progetto di tesi sia durante la realizzazione del progetto stesso. Senza il suo aiuto e i suoi consigli portare a termine questo lavoro in questo modo sarebbe stato sicuramente più difficile.

Infine ci tengo a ringraziare l'ingegnere Arnaud Nguembang Fadja e la professoressa Bellodi per avermi accompagnato nell'ultimo periodo durante il processo di elaborazione di questa tesi.