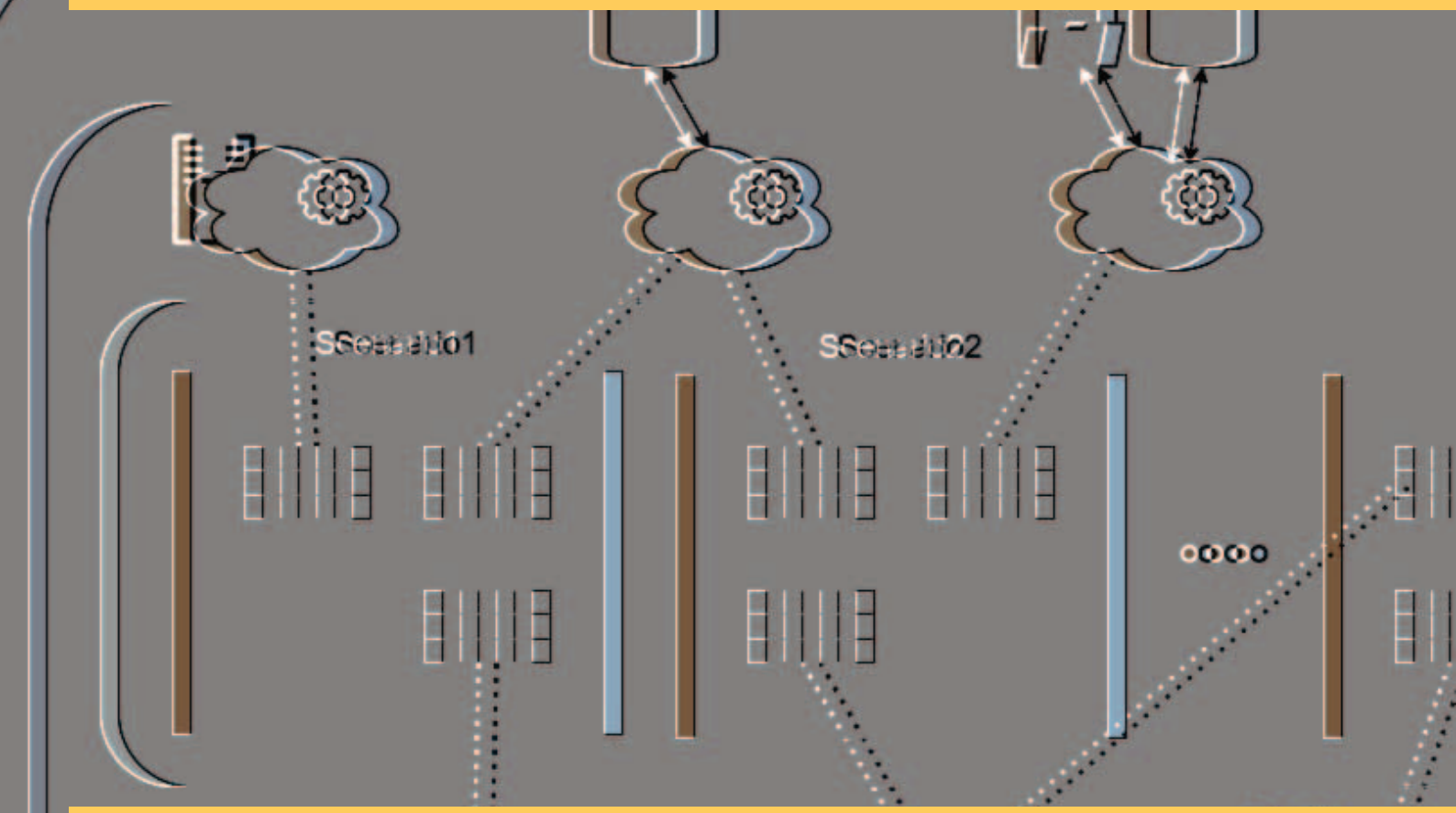
ISSN 2039-7941
anno 2023 numero 462



RAPPORTI TECNICI INGV

SoTableStub: un *tool general purpose* per la visualizzazione di scenari di dati tabellari

ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA

462

REGISTRAZIONE AL TRIBUNALE DI ROMA N.174 | 2014, 23 LUGLIO

ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA

ISSN 2039-7941
anno **2023** numero **462**

RAPPORTI
TECNICI INGV

SoTableStub: un *tool general purpose* per la visualizzazione di scenari di dati tabellari

SoTableStub: a general purpose tool for the visualization of tabular data

Carmelo Cassisi, Marco Aliotta

INGV | Istituto Nazionale di Geofisica e Vulcanologia, Osservatorio Etneo

Accettato 27 settembre 2022 | *Accepted 27 September 2022*

Come citare | *How to cite* Cassisi C., Aliotta M., (2023). *SoTableStub*: un *tool general purpose* per la visualizzazione di scenari di dati tabellari. Rapp. Tec. INGV, 462: 1-30, https://doi.org/10.13127/rpt/462

In copertina Schema esemplificativo della composizione degli scenari gestiti da *SoTableStub*; elaborazione di B. Angioni | *Cover Example diagram of the composition of the scenarios managed by SoTableStub; processed by B. Angioni*

462

INDICE

Riassunto

Durante le attività di ricerca e monitoraggio all'interno dell'Istituto Nazionale di Geofisica e Vulcanologia (INGV-OE) capita sovente che diversi attori manifestino, nel corso del tempo e per scopi differenti, la necessità di consultare particolari sottoinsiemi di dati contenuti in vari database, e di analizzarli in maniera congiunta. Spesso infatti, prima di ricorrere all'uso di sofisticati strumenti di analisi, una mera visualizzazione incrociata dei dati offre un primo e concreto aiuto nell'analisi degli stessi. Ad esempio, si creano continuamente nuove interrogazioni per aggregare e rendere fruibili informazioni relative a gruppi di strumentazioni delle reti di monitoraggio, a particolari serie temporali di interesse o ancora elenchi di servizi appartenenti a sistemi posti in reperibilità. Le interrogazioni di aggregazione, una volta create, vengono poi integrate a strumenti già esistenti (pagine *web*, programmi *client/server*), o talvolta è necessario progettare e implementare *ex novo* strumenti di consultazione dei dati richiesti (che possono a loro volta variare nel corso del tempo). Tali attività, sicuramente ardue (o addirittura proibitive talvolta) da realizzare per un "non addetto ai lavori", impattano sul normale lavoro del personale informatico che, oltre a creare le interrogazioni ai database, deve allo stesso tempo realizzare o integrare di continuo gli strumenti di consultazione dei dati.
L'applicazione *SoTableStub* nasce per rispondere a tale esigenza, permettendo anche agli utenti meno esperti di creare in piena autonomia pannelli di visualizzazione (chiamati "scenari") personalizzabili sia nei contenuti che nello stile, in modo che presentino dati provenienti da diverse fonti e disponibili in formato tabellare. Uno dei suoi punti di forza è il fatto di essere un'applicazione *web*, quindi il suo utilizzo necessita solo di un *web browser*.

Abstract

In the routine activities of research and monitoring within the Istituto Nazionale di Geofisica e Vulcanologia (INGV-OE) different actors manifest, over time and for several purposes, the need to analyse particular subsets of data contained in various databases. Often, in fact, before using sophisticated analysis tools, a mere cross-visualization of the data offers a first and concrete help in the analysis of the same. For example, new queries are constantly being designed to aggregate and provide informations relating to groups of instruments of the monitoring networks or relating to particular time series of interest or even to list services belonging to control room systems. Once created, the aggregation queries are then integrated into existing tools (web pages, client/server applications), or sometimes new tools are designed and implemented from scratch to explore the requested data (which may vary over time). These activities, which are certainly difficult (or even prohibitive) to carry out for a "layman", impact the normal activity of the IT staff who is called to design queries and create new data visualization tools or continuously integrate the existing ones.
The SoTableStub application was created to meet this need, allowing even less experienced users to independently create display panels (called "scenarios") that can be customized both in terms of content and style, in order to show data coming from different sources and available in tabular format. One of its strengths is that it was designed for the web and then users need only a web browser to run the application and visualize data.

Keywords Scenario; Web service; JSON

Introduzione

"Qual è lo stato di acquisizione della rete di monitoraggio? Posso avere degli elenchi distinti per tipologia di sensore o per disciplina?".
"Quali software sono adibiti al monitoraggio dei dati, e quali tra essi risultano offline, e chi sono i destinatari delle allerte relative ai loro malfunzionamenti?".

Queste sono alcune delle domande che possono sorgere nell'ambito della sorveglianza e del monitoraggio (all'INGV e non solo). Per rispondere a tali domande molto spesso bisogna accedere ad informazioni provenienti da diverse fonti, comportando dunque la consultazione di archivi differenti. L'azione di convogliare diversi dati in un'unica rappresentazione di insieme può essere immaginata come la realizzazione di uno "scenario". Una volta formalizzata una richiesta di interesse, essa può diventare il soggetto della creazione di uno scenario. Esso dovrà visualizzare dunque tutte le informazioni che riguardano una particolare tematica. L'applicazione *SoTableStub* nasce con lo scopo di consentire la creazione di scenari personalizzati e ad uso generico, al fine di visualizzare i dati reperiti in forma tabellare.
All'INGV-OE, ad esempio, il sistema *TSDSystem* [Cassisi et al., 2015] che gestisce il database di Sezione contenente i dati vulcanologici, e che alimenta gran parte delle applicazioni sviluppate nell'ambito delle attività di Sala Operativa, necessita di un attento monitoraggio ai fini del suo corretto funzionamento. A tale scopo gli autori del presente report fanno uso di *SoTableStub* sia per ottenere una visualizzazione di insieme dello stato di aggiornamento di tutti i segnali coinvolti nelle attività di sorveglianza, che per condividere in maniera semplice le informazioni principali ai responsabili scientifici/tecnici dei dati che vengono archiviati nel *TSDSystem*. Allo stesso modo si può immaginare uno scenario per rappresentare il quadro generale dello stato dei server su cui girano i software di sorveglianza, e nel quale differenti aspetti, come ad esempio quelli relativi alla loro connettività o alla stima delle risorse di memoria disponibili corrispondono a differenti tipi di interrogazione.
In pratica lo scenario funge da "raccoglitore" di sottoinsiemi di dati che vengono selezionati per "tema". Infatti, certi sottoinsiemi di dati possono essere anche visualizzati in uno o più scenari, sulla base del tema che compongono. L'unico vincolo per la corretta realizzazione di uno scenario è la reperibilità dei dati. Ovviamente è richiesto che i dati seguano uno specifico formato in modo da poter essere gestiti all'interno di uno scenario. I soggetti che si prendono carico di fornire i dati, aderendo a tali requisiti vengono definiti come "servizi", diventando in questo modo gli attori principali di uno scenario. Uno schema esemplificativo di tale logica viene mostrato di seguito in Figura 1.

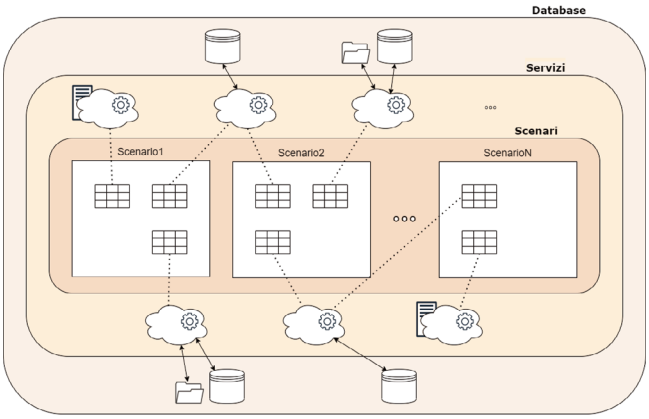


Figura 1 Schema esemplificativo della composizione degli scenari gestiti da *SoTableStub*.
Figure 1 *Example diagram of the composition of the scenarios managed by SoTableStub.*

SoTableStub è un'applicazione web, e come tale la logica del suo funzionamento viene gestita in parte lato *server*, tramite codice seguito all'interno del *server* che ospita l'applicazione, e in parte lato *client*, ovvero dal *web browser* dell'utente che, collegandosi al *server*, riceve il codice da eseguire e ogni sua eventuale risorsa richiesta. Il codice dell'applicazione che viene eseguito lato *server* è stato scritto in *PHP*[1], un linguaggio di *scripting* generico particolarmente adatto allo sviluppo di applicazioni *web*. Il codice che viene eseguito lato *client*, e che si occupa della formattazione delle pagine, delle chiamate al *server* e della creazione delle tabelle, si basa sul linguaggio *JavaScript*. In particolare, per la creazione delle tabelle ci si avvale delle funzionalità di *DataTables*[2], una libreria *JavaScript* che estende la più nota libreria *jQuery*[3], la quale fornisce agli sviluppatori un'interfaccia molto semplice per la manipolazione dei documenti HTML, la gestione degli eventi e le chiamate AJAX. Per l'estetica delle pagine si è scelto di utilizzare le componenti offerte da *Bootstrap*[4], una delle architetture più popolari per la creazione di pagine *web responsive*, ovvero in grado di adattarsi graficamente in modo automatico al dispositivo con cui vengono visualizzate. Tale scelta permette di rendere compatibili le pagine *web* anche per la navigazione su dispositivi *mobile*, migliorando l'accessibilità per gli utenti.
L'applicazione non necessita del supporto di un RDBMS perché le informazioni sulla struttura, nonché sui dati da visualizzare, vengono inserite su un file di configurazione locale, di tipo JSON, chiamato *config.json*, presente nella cartella di progetto[5]. L'unico prerequisito richiesto per istanziare l'applicazione è che il suo codice venga ospitato e gestito da un *web server* predisposto di interprete *PHP* (versione 5.3 o successive).
La struttura del progetto è molto semplice e si basa su due concetti principali, i quali vengono tradotti e trasferiti all'interno del file di configurazione:

- Scenario: uno scenario è inteso come un contenitore che aggrega all'interno di una stessa visualizzazione dati provenienti da diverse fonti.
- Servizio: un servizio è la fonte di un dato.

Il file di configurazione, quindi, contiene un elenco di scenari, ognuno dei quali, a loro volta, contiene l'elenco dei servizi che vengono interrogati per ottenere i dati da visualizzare, con le relative opzioni di visualizzazione.
Al momento della stesura del presente report l'applicazione è in grado di visualizzare dati esposti da servizi in formato JSON o in formato testo (*plain-text*) tabellare, ovvero dove le colonne vengono discriminate attraverso uno specifico carattere di separazione, come ad esempio la virgola nei file CSV.
Le informazioni sulla corretta stesura del file di configurazione sono definite all'interno di un *JSON schema*[6], denominato *schema.json*, presente nella cartella di progetto. Al fine di aiutare l'amministratore dell'applicazione a configurare correttamente il file *config.json*, lo strumento è stato dotato di un validatore, il quale verifica che il file di configurazione segua la struttura definita nello schema JSON. A tale scopo è stata importata nel progetto una libreria *PHP* distribuita sotto licenza MIT su *GitHub* all'indirizzo: https://github.com/justinrainbow/json-schema. La licenza MIT è una licenza di software libero che permette il riutilizzo in un altro software sotto la condizione che la licenza sia distribuita con tale software.

[1] Vedi Sitografia [6]
[2] Vedi Sitografia [2]
[3] Vedi Sitografia [3]
[4] Vedi Sitografia [1]
[5] Il progetto è reso pubblico all'indirizzo: https://github.com/carmelocassisi/SoTableStub
[6] Vedi Sitografia [4]

1. Gli scenari

Ogni scenario può essere visualizzato chiamando il seguente indirizzo:

http://*[myWebServerAddress]*/**SoTableStub**/*(index.php)*?**scenario**=*[myScenarioID]*

dove [*myWebServerAddress*] rappresenta l'indirizzo del *server web* che ospita l'istanza dell'applicazione *SoTableStub*, e [*myScenarioID*] l'identificativo dello scenario desiderato. Se si vuole evitare di scrivere *index.php* nell'indirizzo, allora sul *web server* che gestisce l'applicazione deve essere impostata la regola con la quale si aggiunge *index.php* tra le pagine predefinite. Se l'identificativo dello scenario non viene indicato, verrà caricato lo scenario predefinito contenuto nel file *config.json* con "*ID*": *"HOME"* descritto nel paragrafo 1.2.
Per i dettagli sulla struttura e sulla configurazione degli scenari si rimanda al Capitolo 3.

1.1 Visualizzazione di uno scenario

Per mostrare la struttura di una pagina corrispondente a uno scenario prendiamo come esempio lo scenario indicato nel file di configurazione con identificativo "*ID*": "*SO_Checker*" (Figura 2).
Nella parte superiore della pagina si trova un'intestazione contenente il titolo e la descrizione dello scenario, indicati rispettivamente dai valori assegnati ai campi "*Title*" e "*Description*" nel file di configurazione.
Il corpo della pagina viene invece popolato dai servizi richiesti dallo scenario. Ogni servizio viene rappresentato da un titolo e un sottotitolo (indicati rispettivamente dai valori assegnati ai campi "*Name*" e "*Description*" di ogni servizio nel file di configurazione), seguito da una tabella che ne visualizza la risorsa fornita. Nella visualizzazione di default, nel caso in cui uno scenario interroghi più servizi, questi vengono visualizzati uno di seguito all'altro.
Stessa struttura viene preservata nella visualizzazione su dispositivo mobile (Figura 3), dove gli elementi della pagina vengono adattati alle dimensioni dello schermo utilizzato grazie alle funzionalità offerte dalle librerie *Bootstrap* e *DataTables*.
È possibile impostare una visualizzazione "a schede" dello scenario, agendo sul file di configurazione (in particolare, settando l'attributo *"makeTabs": true* tra le *options* dello scenario), in modo che ciascun servizio afferente allo scenario venga mostrato in un'apposita scheda, etichettata dal proprio *ID*. La Figura 4 mostra l'esempio di uno scenario dal titolo "*SO Checker*", il quale interroga due servizi con "*ID*": "*SODiskChecker*" e "*ID*": "*SOPingChecker*" (scheda attiva).
Se si utilizza la visualizzazione a schede, allora è possibile indicare nella pagina dello scenario anche quale servizio mostrare in primo piano (rendendo attiva la scheda corrispondente, Figura 4). Per fare ciò basta aggiungere nella *querystring* il parametro *service*, passando come valore *l'ID* del servizio:

http://*[myWebServerAddress]*/**SoTableStub**/*(index.php)*?**scenario**=*[myScenarioID]*&**service**=*[myServiceID]*

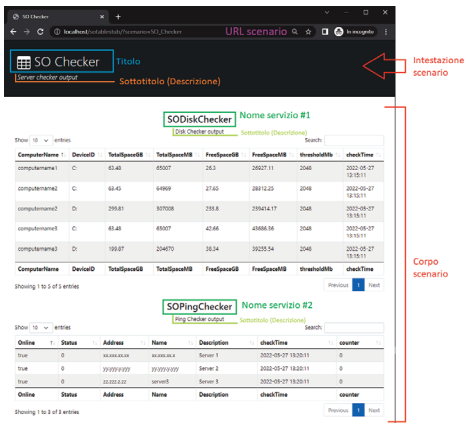


Figura 2 Struttura di una pagina corrispondente allo scenario con "*ID*": "*SO_Checker*". http://[*myWebServerAddress*]/SoTableStub/?scenario=SO_Checker

Figure 2 *Structure of a page related to the scenario with "ID": "SO_Checker". http://[myWebServerAddress]/SoTableStub/?scenario=SO_Checker*

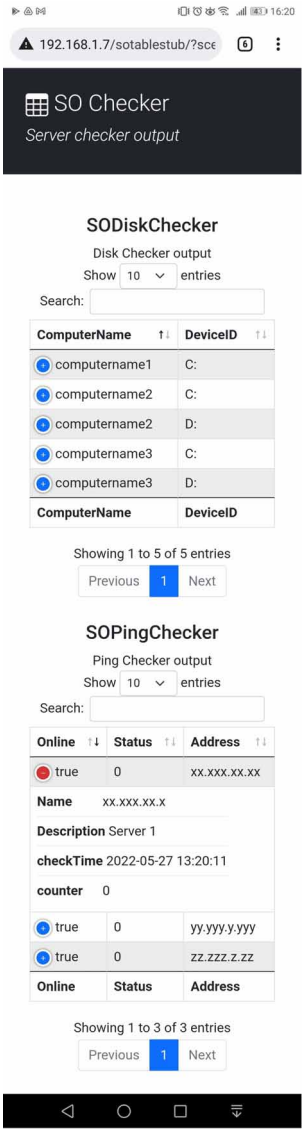


Figura 3 Struttura della pagina dello scenario con "*ID*": "*SO_Checker*" su dispositivo mobile.

Figure 3 *Structure of the page on a mobile device for the scenario with "ID": "SO_Checker".*

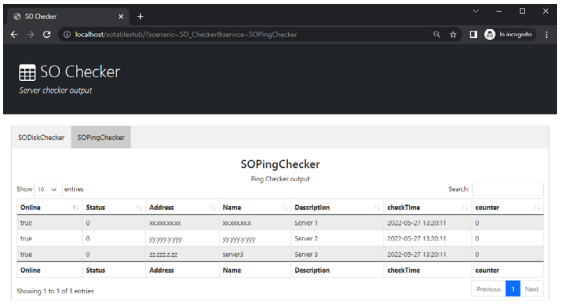


Figura 4 Struttura dello scenario dal titolo "*SO Checker*" in modalità "visualizzazione a schede".

Figure 4 *Structure of a scenario entitled "SO Checker" in "tab view" mode.*

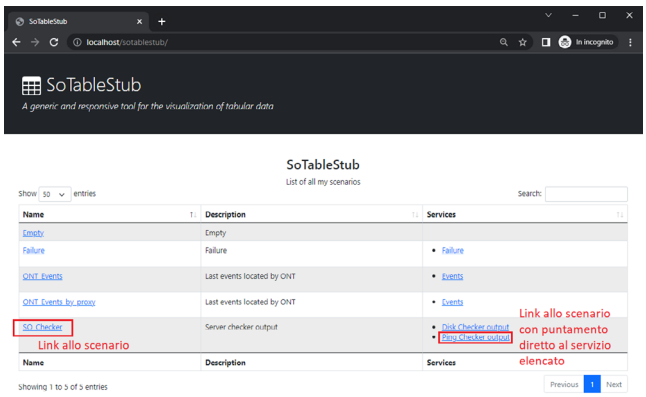
1.2 La pagina principale (HOME)

La pagina principale non è altro che la visualizzazione dello scenario predefinito con "*ID*": "*HOME*". Di seguito un estratto del file *config.json* di progetto, rappresentante il suddetto scenario.

```
{
	"ID": "HOME",
	"Title": "SoTableStub",
	"Description": "A generic and responsive tool for the …",
	"private": true,
	"services": [{
		"ID": "SoTableStub",
		"Name": "SoTableStub",
		"Description": "List of all my scenarios",
		"URL": ".\\Services\\MyScenarios.php",
		"table_options": {
			"responsive": true,
			"pageLength": 50,
			"order": [
				[0, "asc"]
			]
		}
	}]
}
```

Lo scenario con "*ID*": "*HOME*" è costituito da un unico servizio implementato dallo script *MyScenarios.php*, contenuto all'interno della sottocartella di progetto *Services*, il quale restituisce la lista degli scenari disponibili tra quelli contenuti nel file *config.json* (Tabella in Figura 5).
Per discriminare quali scenari verranno elencati nella tabella della pagina principale (ad es. per non elencare lo scenario relativo alla HOME stessa, evitando un puntamento circolare), si può fare uso della proprietà *private* dello scenario, di tipo booleano. Se nel file di configurazione, nella definizione di uno scenario, viene impostata la proprietà "*private*": *true*, allora lo scenario non verrà elencato tra quelli disponibili nella pagina principale.

Figura 5 Visualizzazione della pagina principale, ovvero dello scenario con "*ID*": "*HOME*".
Figure 5 *Snapshot of the main page, that is the scenario with "ID": "HOME".*



Qualora non venga indicato lo scenario con "*ID*": "*HOME*" all'interno del file *config.json*, la pagina principale mostrerà un errore.

Nel caso in cui il file di configurazione non sia correttamente compilato, la pagina principale non mostrerà la tabella degli scenari disponibili, ma un messaggio di errore contenente le informazioni sulle violazioni riscontrate dal validatore rispetto allo schema JSON (Figura 6). *JSONSchemaValidation.php* è il file di progetto contenente il codice che esegue tale validazione. Potrebbe anche capitare che all'interno del file di configurazione possano essere stati utilizzati gli stessi identificativi (proprietà *ID*) per degli scenari e/o per dei servizi all'interno di uno stesso scenario. In tal caso, la pagina principale mostrerà un messaggio di avviso (Figura 7) che elenca i duplicati trovati all'interno del file di configurazione, che potrebbero causare delle anomalie nella manipolazione degli elementi del DOM. In particolare, il testo dell'errore mostra solo il primo duplicato riscontrato.
A proposito dei duplicati c'è da dire che, anche se lo schema JSON (si veda in Appendice A - Lo schema JSON, la definizione dello schema JSON) prevede la proprietà *"uniqueItems": true* sia per *l'array* degli scenari che per *l'array* dei servizi all'interno di uno scenario, si è reso necessario aggiungere un controllo *custom* sugli *ID* duplicati in quanto, secondo l'attuale *release* (2020-12, al momento della stesura del report), la proprietà *uniqueItems* effettua un controllo sull'intero elemento, sia esso un oggetto di tipo base (come ad es. numeri o stringhe) che complesso (come ad es. gli array o gli oggetti). Quindi l'unicità di un elemento non può essere determinata solo da una proprietà, anche se ritenuta identificativa, ma dall'intero contenuto dell'oggetto che lo rappresenta. La verifica di unicità, pertanto, fallisce solo se due scenari o servizi sono uguali in OGNI sua parte, compreso l'*ID*.

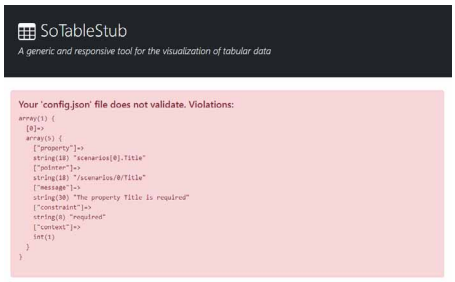


Figura 6 Errore mostrato nella pagina principale nel caso di un'errata compilazione del file *config.json*.
Figure 6 *Error message shown on the main page in case of an incorrect compilation of the config.json file.*

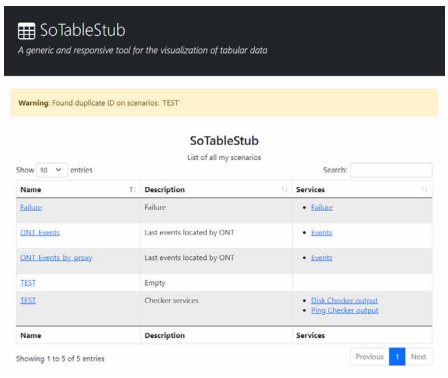


Figura 7 Messaggio di avviso mostrato nella pagina principale in caso di individuazione di ID duplicati.
Figure 7 *Warning message shown on the main page in case of identification of duplicated IDs.*

2. I servizi

I dati di ogni servizio vengono recuperati dall'applicazione *web* attraverso delle chiamate HTTP GET asincrone (tramite il paradigma AJAX) alla fonte del dato stesso, ovvero all'URL del servizio. Come già specificato, al momento vengono gestiti servizi che rispondono con dati in formato JSON o testuale. Nella cartella di progetto, i servizi di esempio si trovano all'interno della sottocartella *Services*. In questa cartella gli amministratori del sistema possono inserire altri script che fanno da *proxy* per servizi non direttamente interrogabili dal *web browser*, o per il cui risultato si vogliano aggiungere delle personalizzazioni grafiche.

2.1 Formati di dato gestiti

2.1.1 JSON

Di seguito un esempio di *dataset* in formato JSON.

```
[
    {
        “col_1”: <value>,
        “col_2”: <value>,
        …
        “col_n”: <value>
    },
  …
    {
        “col_1”: <value>,
        “col_2”: <value>,
        …
        “col_n”: <value>
    }
]
```

2.1.2 Testo (CSV o similare)

Di seguito un esempio di *dataset* in formato testo (nel CSV le colonne sono separate da virgole).

```
col_1, col_2, …, col_n
<value>, <value>, … <value>
<value>, <value>, … <value>
…
<value>, <value>, … <value>
```

2.2 Esempi di servizio

2.2.1 MyScenarios.php

L’esempio principale è lo script *MyScenarios.php*, distribuito insieme al codice dell’applicazione nella sottocartella di progetto *Services*, ed interrogato dallo scenario con “*ID*”: “*HOME*”. Lo script implementa il servizio che restituisce la lista di tutti gli scenari da visualizzare nella pagina principale (Figura 5). Ogni elemento della lista contiene il nome dello scenario (colonna *Name*), la descrizione (colonna *Description*) e i servizi associati allo scenario (colonna *Services*). Per aggiungere funzionalità alla tabella risultante, il servizio *MyScenarios.php* “decora” il nome di ogni scenario dotandolo del *link* alla relativa pagina; inoltre, i servizi associati ad ogni scenario vengono rappresentati sotto forma di elenco puntato, con ciascun punto elenco recante il nome del servizio, a sua volta “decorato” con il *link* diretto alla posizione (eventualmente la scheda) in cui viene visualizzato all’interno del corpo dello scenario.

2.2.2 Lista in tempo reale degli eventi localizzati dall'ONT

Lo scenario con "*ID*": "*Events*" interroga il *web service dell'Osservatorio Nazionale Terremoti* (ONT)[7], in particolare l'API *fdsnws-event* del catalogo *ISIDe* [ISIDe Working Group, 2007] per ottenere la lista degli ultimi eventi localizzati (Figura 8). La configurazione del servizio da includere nello scenario potrebbe essere la seguente:

```
{
    "ID": "Events",
    "Name": "Events",
    "Description": "Events",
    "URL": "http://webservices.rm.ingv.it/fdsnws/event/1/query?format=text",
    "dataType": "text",
    "textSeparator": "|"
}
```

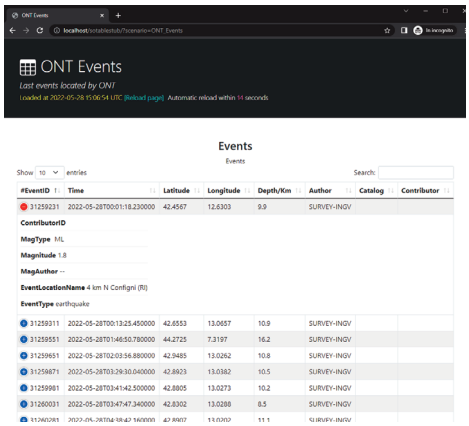


Figura 8 Tabella risultante dall'interrogazione del web service dell'ONT: http://webservices.rm.ingv.it/fdsnws/event/1/query?format=text

Figure 8 *Table rendered by querying the ONT web service: http://webservices.rm.ingv.it/fdsnws/event/1/query?format=text*

Come si può notare in Figura 8, grazie all'utilizzo di *Bootstrap* e *DataTables*, le colonne che non riescono ad essere visualizzate orizzontalmente vengono raggruppate e rese visualizzabili verticalmente premendo sul pulsante "+" all'inizio della riga.

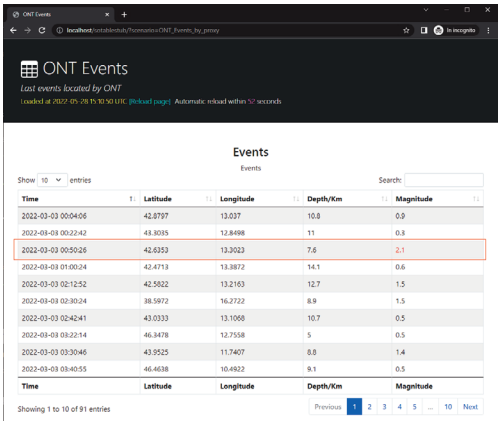
2.2.3 Servizi personalizzati

Possono nascere delle esigenze in base alle quali si vogliano aggiungere personalizzazioni nella grafica relativa alle tabelle. Prendendo ad esempio lo scenario degli ultimi eventi localizzati dall'ONT, si potrebbe voler colorare in rosso il valore relativo alla magnitudo per gli eventi con magnitudo maggiore di 2, visualizzando possibilmente solo alcune colonne di interesse, o formattando le date in un certo modo (Figura 9).
Per realizzare ciò, è possibile scrivere uno script *proxy* e far puntare ad esso piuttosto che puntare direttamente all'URL del *web service*. Gli script *proxy* possono ritornare utili anche quando l'interrogazione di un *web service* richiede una fase di autenticazione o l'utilizzo di *token* autorizzativi, che possono essere programmati all'interno dello *script* stesso. Lo scopo dello script *proxy* che interroga il *web service* è quello di elaborare il risultato e ricostruire il *dataset* in modo da ottenere il risultato desiderato. Nel caso specifico si dovranno eliminare le colonne non pertinenti e inglobare il valore della magnitudo in un *tag* HTML a cui viene applicato uno stile

[7] Vedi Sitografia [5]

CSS, aggiungendo ad esempio l'attributo *style = "color: red"*, oppure assegnando delle classi già predisposte da *Bootstrap* (ad es. aggiungendo l'attributo *class = "text-danger"*), sfruttando le potenzialità di personalizzazione dell'applicazione.
Di seguito un'implementazione in PHP (disponibile anche nella sottocartella di progetto *Services* con nome *proxy_events_by_curl_custom_view.php*).

Figura 9 Esempio di servizio personalizzato.
***Figure 9** Example of customized service.*




```php
<?php
$url = "http://webservices.rm.ingv.it/fdsnws/event/1/query?format=text";
$col_sep = "|";

$ch = curl_init(); // create curl resource
curl_setopt($ch, CURLOPT_URL, $url); // set url
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1); //return the transfer as a string
$output = curl_exec($ch); // $output contains the output string
curl_close($ch); // close curl resource to free up system resources

// split result lines and load into array
$result = preg_split("/\r\n|\n|\r/", $output);

// exit if empty result
if (count($result) < 1) exit;

// select headers
$headers = explode($col_sep, $result[0]);

// customization: columns we don't want to see
$disabled_headers = array(
    "#EventID",
    "Author",
    "Catalog",
    "Contributor",
    "ContributorID",
    "MagType",
    "MagAuthor",
    "EventLocationName",
    "EventType"
);
```

```
$toJSON = array();
for($i=1; $i<count($result)-1; $i++) {
    $row = explode($col_sep, $result[$i]);
    $toJSONrow = array();
    for($j=0; $j<count($headers); $j++) {
        if (!in_array($headers[$j], $disabled_headers)) {
            $value = isset($row[$j]) ? $row[$j] : '';
            $toJSONrow[$headers[$j]] = $value;

            // customization
            if ($headers[$j] == "Magnitude" and floatval($row[$j]) > 2) {
                    $toJSONrow[$headers[$j]] = "<span class="font-weight-bold text-
danger">" . $value . "</span>"; // red color for events with mag > 2
            }
            if ($headers[$j] == "Time") {
                                $toJSONrow[$headers[$j]] = str_replace("T", " ",
substr($toJSONrow[$headers[$j]], 0, 19)); // customize time format
            }
        }
    }
    array_push($toJSON, $toJSONrow);
}

echo json_encode($toJSON, JSON_NUMERIC_CHECK);

?>
```

2.2.4 Servizi non disponibili

La Figura 10 mostra un esempio di servizio il cui risultato non è nel formato accettato dall'applicazione, e che viene gestito mostrando all'utente il messaggio di tipo *Incorrectly formatted dataset*.
Ogni tipo di chiamata HTTP non andata a buon fine viene gestita nelle stesse modalità, ovvero mostrando il messaggio di errore all'interno del riquadro rosso, al posto della tabella attesa. Ad esempio, nel caso in cui l'URL del servizio non esistesse o non sia disponibile, il messaggio di errore conterrà la dicitura *Not Found*.

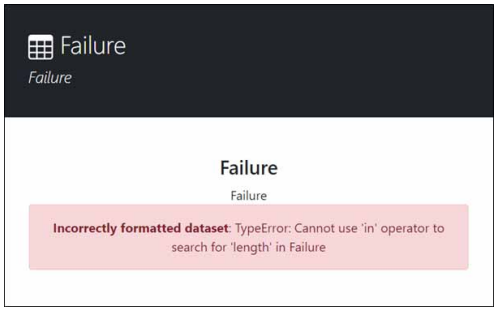


Figura 10 Esempio di servizio in cui il risultato non viene restituito nel formato atteso.
Figure 10 *Example of a service where the result is not in the expected format.*

2.3 Cross-Origin Resource Sharing

Questa applicazione può diventare a tutti gli effetti un'implementazione di *Cross-Origin Resource Sharing* (CORS), ovvero di condivisione di risorse tra origini diverse (Figura 11). CORS è un meccanismo basato sulle intestazioni HTTP, e consente a un *server* di indicare qualsiasi origine (dominio, schema o porta) diversa dalla propria, da cui un *web browser* dovrebbe consentire il caricamento delle risorse.

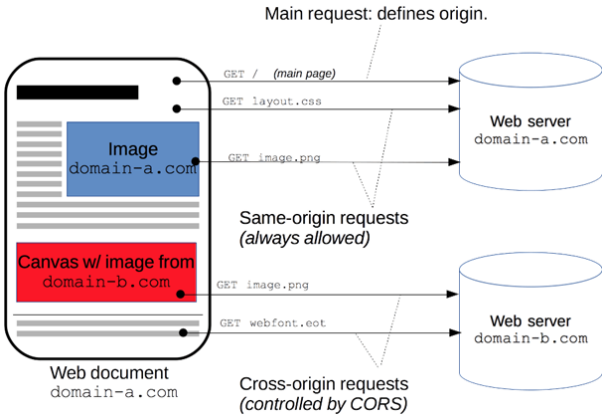


Figura 11 Schema esemplificativo di una pagina web popolata da informazioni/risorse provenienti da diverse origini, costituendo di fatto le dinamiche del meccanismo CORS (*Cross-Origin Resource Sharing, CORS*, https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS).

Figure 11 *Example scheme showing a web page populated by informations/resources coming from different origins, constituting the dynamics of the CORS mechanism (Cross-Origin Resource Sharing, CORS, https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS).*

Per motivi di sicurezza, i *web browser* limitano le richieste HTTP multi-origine avviate dagli *script*. Infatti, uno dei principali motivi per cui una chiamata ad un servizio può fallire riguarda la cosiddetta *CORS policy*.

Ad esempio, *XMLHttpRequest*, che è l'API *web* utilizzata dal linguaggio *JavaScript* (e quindi anche dalla nostra applicazione) per effettuare tali richieste, aderisce a questa politica. Un tipico messaggio di errore restituito dalla libreria sulla base dell'applicazione di questa *policy* è il seguente: *"Access to XMLHttpRequest at <myServiceURL> from origin http://[myWebServerAddress] has been blocked by CORS policy: No Access-Control-Allow-Origin header is present on the requested resource".*

Ciò significa che *XMLHttpRequest* può richiedere risorse solo dalla stessa origine da cui è stata caricata l'applicazione, a meno che la risposta da altre origini non includa le intestazioni *CORS* corrette. *MyScenarios.php*, ad esempio, non risente di questo controllo perché è un servizio che gira sullo stesso *web server* dell'applicazione. Quindi, per fare in modo che un servizio possa essere utilizzato in uno scenario e visualizzato senza che il *web browser* ne impedisca la richiesta, il *server* che fornisce la risorsa deve consentirlo esplicitamente.

Se, ad esempio, il dato viene generato lato *server* da uno script *PHP*, allora tale script dovrà contenere la seguente riga di codice: *header(Access-Control-Allow-Origin: *)*. Così facendo, nell'intestazione della risposta HTTP verrà aggiunta la voce *Access-Control-Allow-Origin* con valore "*" che indica: *consentire a tutti.*

Se invece, ad esempio, la risorsa è un file JSON ospitato su un *web server* IIS (*Internet Information Services*), allora dall'interfaccia di gestione di IIS è necessario selezionare la cartella dove risiede la risorsa e impostare alla voce *Intestazioni risposte http* il valore *Access-Allow-Control-Origin=**.

Una politica più prudente e mirata consiste nello specificare, al posto del valore "*", l'indirizzo IP del *server* a cui si vuole consentire l'accesso (nel nostro caso la macchina *server* dove gira l'applicazione *SoTableStub* e dal quale arrivano le richieste), escludendo così eventuali chiamate da altri indirizzi IP.

3. Il file di configurazione

Il file di configurazione è il cuore dell'applicazione e l'unico file che un utente amministratore deve saper manipolare al fine di ottenere le visualizzazioni desiderate.
Il file *config.json* rappresenta l'oggetto principale *Scenarios*, descritto da una sola proprietà *scenarios* obbligatoria (di tipo *array*), in cui vanno elencate tutte le istanze dell'oggetto *Scenario*.

3.1 Struttura Scenario

Ogni *Scenario* è descritto dalle seguenti proprietà (* indica una proprietà obbligatoria):
*ID: string**
*Title: string**
Description: string
private: boolean
options: object

Nella Figura 12 viene riportato il diagramma dello schema JSON utilizzato nel file di configurazione, dove nella parte destra viene riportata la struttura di uno scenario. Le prime tre proprietà (*ID*, *Title*, *Description*) sono auto-esplicative, poiché corrispondono rispettivamente all'identificativo dello scenario, al titolo che si vuole mostrare nella pagina dello scenario e alla sua descrizione che verrà mostrata come sottotitolo. La proprietà *private* indica se si vuole che lo scenario venga elencato tra gli scenari della pagina principale o meno.

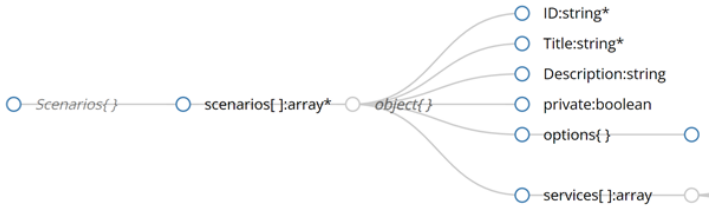


Figura 12 Diagramma dello schema JSON utilizzato per il file di configurazione.
Figure 12 *Diagram of the JSON schema used for the configuration file.*

Per quanto riguarda *options* (Figura 13), di seguito le proprietà configurabili e gestite all'interno del codice:

- *makeTabs* (*boolean*), permette di selezionare una visualizzazione a schede per le tabelle di uno scenario;
- *showCurrentTimeUTC* (*boolean*), aggiunge nell'intestazione della pagina dello scenario l'orario di caricamento dei dati mostrati in tabella (utile per visualizzazioni di dati che vengono aggiornati in tempo reale), vedi Figura 14;
- *refreshInterval* (*integer*), accetta valori interi maggiori di zero e indica l'intervallo (espresso in secondi) dopo il quale la pagina dello scenario verrà ricaricata automaticamente. Il conto alla rovescia dei secondi rimanenti al successivo *refresh* viene indicato nell'intestazione della pagina dello scenario (Figura 14).

All'interno della sezione *Services* di uno scenario vanno elencate tutte le istanze di tipo *Service* a cui si aggancia lo scenario per ottenere i dati.

Figura 13 Dettaglio della struttura dello scenario all'interno dello schema JSON.
Figure 13 *Detail of the structure of the scenario within the JSON schema.*

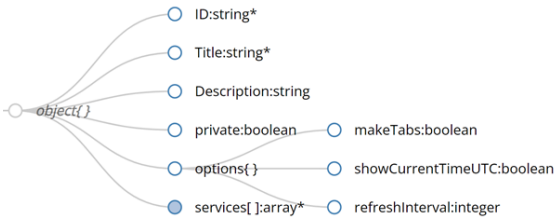
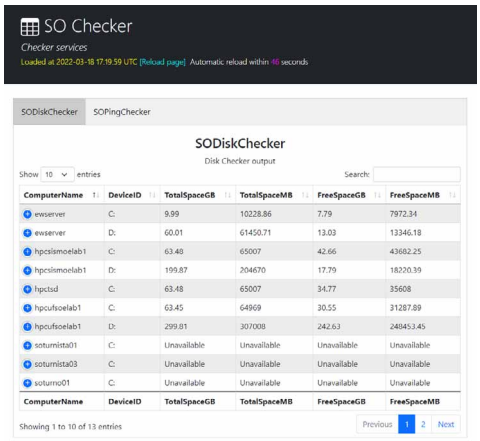


Figura 14 Esempio dello scenario dal titolo "*SO Checker*", dove sono state abilitate le opzioni *showCurrentTimeUTC* e *refreshInterval*.
Figure 14 *Example of a scenario entitled "SO Checker" where the showCurrentTimeUTC and refreshInterval property have been enabled.*



3.2 Struttura Service

Ogni *Service* è descritto dalle seguenti proprietà (* indica una proprietà obbligatoria):
*ID: string**
*Name: string**
URL: string
dataType: string
textSeparator: string
table_options: object

Nella Figura 15 viene riportato il diagramma dello schema JSON utilizzato nel file di configurazione, dove nella parte destra viene riportato in dettaglio la struttura del servizio. Le proprietà *ID*, *Name*, e *Description* indicano rispettivamente l'identificativo del servizio, il nome/titolo che si vuole mostrare sopra la tabella associata ai dati restituiti dal servizio, e la sua descrizione che verrà mostrata come sottotitolo. La proprietà *URL* indica la fonte dei dati restituiti dal servizio. Il formato predefinito accettato è il JSON, ma se il servizio fornisce un tipo di dato testuale (sempre in forma tabellare), allora si deve impostare la proprietà "*dataType*"**:** "*text*", indicando eventualmente anche, tramite la proprietà *textSeparator*, il carattere che funge da separatore di colonna (di *default* viene considerata la virgola).
La sezione *table_options* è invece dedicata a *DataTables*, la libreria *JavaScript* utilizzata per costruire le tabelle contenenti i dati; tale libreria aggiunge diverse funzionalità, tra le quali: paginazione per una migliore navigazione dei dataset contenenti un numero elevato di record, filtraggio delle righe tramite ricerca testuale, ordinamento dei valori per colonna. Poiché la configurazione di tali funzionalità viene espressa nel formato JSON, *SOTableStub* si limiterà a inoltrare la sezione *table_options* di un servizio al costruttore della classe *JavaScript* che

implementa la tabella da visualizzare. Per la compilazione di tale sezione si rimanda all'indirizzo: https://datatables.net/manual/options.

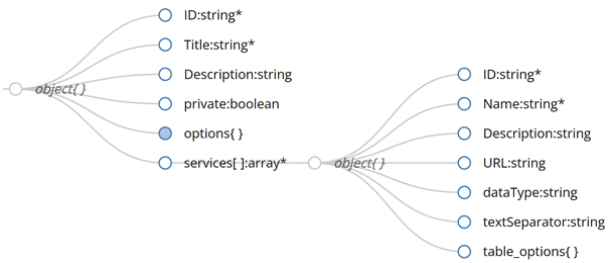


Figura 15 Dettaglio della struttura del servizio all'interno dello schema JSON.
Figure 15 *Detail of the structure of the service within the JSON schema.*

Per ogni altro dettaglio riguardante la struttura del file di configurazione di *SOTableStub* si rimanda alla definizione integrale dello schema JSON in Appendice A - Lo schema JSON.

4. Conclusioni

Il nostro progetto si presta non solo ad una moltitudine di applicazioni, ma può abbracciare diversi tipi di utenze. Chi non possiede conoscenze di programmazione, ma conosce le URL dei servizi dai quali reperire i dati, ha a disposizione uno strumento flessibile, che gli consente di creare scenari di visualizzazione su tabella, compilando solamente un file di configurazione JSON. In più, il prodotto è finalizzato sia per un utilizzo esclusivamente personale, sia per una condivisione attraverso il *web*.
Gli utenti più esperti, invece, hanno a disposizione un prodotto pronto all'uso, ma anche altamente personalizzabile. L'aspetto più importante di tale personalizzazione è l'assenza di un intervento sul codice principale dell'applicazione. Infatti, tutto può essere pilotato attraverso il file di configurazione (incluse le funzionalità offerte dalla libreria *DataTables*, attraverso la compilazione della relativa sezione) o tramite la creazione di terze parti, sviluppate nel linguaggio e nel contesto più familiare allo sviluppatore, richiamabili come servizi e che possono adeguarsi a specifiche esigenze di visualizzazione e/o autorizzazioni, sfruttando anche gli strumenti di stile offerti dal CSS, nonché da *Bootstrap*.

Il progetto viene distribuito come software libero all'indirizzo:
https://github.com/carmelocassisi/SoTableStub

Bibliografia

Cassisi C., Montalto P., Aliotta M., Cannata A., Prestifilippo M. (2015). *TSDSystem: un database multidisciplinare per la gestione di serie temporali*. Rapporti Tecnici INGV, 304: 1-34, ISSN: 2039-7941.

ISIDe Working Group, (2007). *Italian Seismological Instrumental and Parametric Database (ISIDe)*. Istituto Nazionale di Geofisica e Vulcanologia (INGV), https://doi.org/10.13127/ISIDE

Sitografia

[1] *Bootstrap*. https://getbootstrap.com/
[2] *DataTables*. https://datatables.net/

[3] *jQuery*. https://jquery.com/
[4] *JSON Schema*. https://json-schema.org/
[5] *ONT Web Service*. http://webservices.ingv.it/fdsnws/event/1/
[6] *PHP*. https://www.php.net/

Glossario

AJAX, *Asynchronous JavaScript And XML*, una raccolta di diverse tecnologie volte a fornire una migliore esperienza utente rispetto alle tradizionali applicazioni *web*. Il suo utilizzo permette agli sviluppatori di aggiornare una pagina *web* senza ricaricare la pagina nel suo insieme (ma solo in parte), ricevere dati da un *server* dopo che la pagina è stata caricata, e/o inviare dati a un *server* in *background* (ovvero in maniera inconsapevole per l'utente).

API, *Application Programming Interface*, insieme di procedure per l'espletamento di un dato compito; spesso tale termine designa le librerie software di un linguaggio di programmazione.

client, nelle architetture di reti informatiche indica un terminale che si connette ad un *server* per la fruizione di un certo servizio o risorsa.

CORS, *Cross-Origin Resource Sharing*, è un meccanismo che consente di richiedere risorse limitate su una pagina *web* da un dominio esterno rispetto al dominio da cui viene servita la pagina.

CSS, *Cascading Style Sheets*, un linguaggio utilizzato per la formattazione dei documenti HTML.

CSV, *Comma Separated Values*, è un formato file di tipo testuale, organizzato per colonne, dove il carattere utilizzato per la separazione tra le colonne è la virgola.

dataset, collezioni di dati.

DOM, *Document Object Model*, è una forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti. Nativamente supportato dai *web browser* per modificare gli elementi di un documento HTML, DOM è un modo per accedere e aggiornare dinamicamente il contenuto, la struttura e lo stile dei documenti.

fdsnws-event, *web service* le cui specifiche sono state formalizzate dall'IRIS (*Incorporated Research Institutions for Seismology*) e utilizzate come standard per la consultazione di database di eventi sismici.

hosting, in informatica indica un servizio che ospita pagine *web* da rendere disponibili per l'interrogazione tramite l'utilizzo di protocolli di rete (HTTP, FTP, etc.).

indirizzo IP, codice numerico che identifica un dispositivo registrato sulla rete Internet e attraverso il quale si sviluppa il protocollo che si occupa dell'instradamento dei pacchetti di comunicazione (*Internet Protocol*).

GitHub, un servizio di *hosting* per progetti software.

HTML, *HyperText Markup Language*, è un linguaggio di *markup* nato per la formattazione e l'impaginazione di documenti ipertestuali.

HTTP, *HyperText Transfer Protocol*, è un protocollo usato come principale sistema per la trasmissione d'informazioni sul *web*. Le specifiche del protocollo sono gestite dal W3C.

HTTP *GET*, metodo del protocollo HTTP, designato alla richiesta di dati per la visualizzazione.

HTTPS, *HyperText Transfer Protocol over Secure Socket Layer*, protocollo HTTP cifrato tramite protocollo TLS o SSL.

JavaScript, un linguaggio di *scripting* interpretato, comunemente utilizzato nella programmazione *web* lato *client* (esteso poi anche al lato *server*) per la creazione di pagine *web* dinamiche.

JSON, *JavaScript Object Notation*, è un formato adatto all'interscambio di dati fra applicazioni *client/server* nell'ambito della programmazione *web*.

JSON *Schema*, è un vocabolario che consente di definire la struttura di un documento JSON, nonché convalidarlo.

link, in un ipertesto è un testo o un'immagine che funge anche da collegamento ad un altro

elemento dello stesso testo o ad una risorsa del *web*.

plain-text, è un documento che contiene solamente testo puro, ossia la codifica binaria di caratteri comprensibili a un lettore umano (come lettere, numeri, segni di punteggiatura, ecc.).

PHP, acronimo ricorsivo di *PHP*: *Hypertext Preprocessor*, linguaggio di *scripting* interpretato, concepito per la programmazione di pagine *web* dinamiche.

proxy, in informatica e telecomunicazioni, indica un soggetto che funge da intermediario per le richieste da parte dei client alla ricerca di risorse su altri server.

querystring, letteralmente "stringa di ricerca", è la parte di un URL che contiene dei dati da passare in input ad una pagina *web* (di solito si trovano alla fine della stringa dell'URL, subito dopo il carattere speciale "?").

RDBMS, *Relational DataBase Management System*, un insieme di tecnologie e software per la gestione e l'interrogazione di basi di dati relazionali.

responsive, nell'ambito del *web design* è l'attitudine di una pagina *web* di adattare graficamente i suoi contenuti in modo automatico al dispositivo col quale vengono visualizzati (ad es. computer con diverse risoluzioni, tablet, smartphone), migliorando l'accessibilità per gli utenti.

release, in informatica, ciascuna nuova versione di un software, o di una specifica messa in commercio o comunque diffusa, contraddistinta da un identificativo alfanumerico.

scripting, un linguaggio di *scripting* è un linguaggio di programmazione che esegue attività all'interno di uno speciale ambiente di esecuzione da un software interprete anziché da un compilatore.

server, nelle architetture di reti informatiche indica un computer o terminale che fornisce un certo servizio o risorsa ed è contattabile da altri terminali *client*.

***tag* HTML**, sono gli elementi del linguaggio HTML che permettono di scrivere un documento in formato HTML, organizzato in una struttura logicogerarchica consultabile via *web browser*.

***token* ASCII**, insieme di caratteri codificati attraverso lo standard ASCII (*American Standard Code for Information Interchange*) che non contiene spazi vuoti (es. spazi, tabulazioni) o caratteri speciali (inizio/fine riga).

URL, *Uniform Resource Locator*, una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa nella rete (locale, interna o internet).

W3C, *World Wide Web Consortium*, è una comunità internazionale finanziata sia da organizzazioni membro che da fondi pubblici e privati, nonché da donazioni da parte di singoli individui, dove personale dedicato e finanziatori collaborano insieme per lo sviluppo degli standard per il *web*.

web, sottorete di Internet per il trasferimento e la visualizzazione di dati nel formato ipertestuale, di solito HTML.

web browser, software per l'acquisizione, la presentazione e la navigazione di risorse, come pagine *web*, immagini o video, disponibili su rete informatica tramite protocollo HTTP o HTTPS.

web service, secondo la definizione del W3C all'indirizzo https://www.w3.org/TR/ws-arch/, è un sistema software progettato per supportare l'interazione macchina-macchina attraverso una rete informatica.

Appendice A – Lo schema JSON

Nello schema JSON, per gli identificativi degli scenari e dei servizi, si è scelto di avvalersi di *token* ASCII che possano essere utilizzati come attributi identificativi (*id*) dei *tag* HTML[8] che compongono le pagine visualizzate, in modo da poter essere identificabili e manipolabili attraverso il DOM. Di seguito il contenuto del file *schema.json*.

```
{
    "$id": "http://[myWebServerAddress]/SoTableStub/schema.json",
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "title": "Scenarios",
    "description": "The configuration file schema for SoTableStub",
    "type": "object",
    "required": ["scenarios"],
    "properties": {
      "scenarios": {
        "type": "array",
        "description": "The list of scenarios.",
        "items": { "$ref": "#/$defs/scenario" },
        "uniqueItems" : true
      }
    },
    "$defs": {
        "scenario": {
          "type": "object",
          "required": [ "ID", "Title" ],
          "properties": {
                "ID": {
                    "type": "string",
                    "pattern": "^[a-zA-Z0-9_]+$",
                    "description": "The identificative name of the scenario."
                },
                "Title": {
                    "type": "string",
                    "description": "The title of the scenario."
                },
                "Description": {
                    "type": "string",
                    "description": "The description of the scenario."
                },
                "private": {
                    "type": "boolean",
                    "description": "Set the scenario visible or not into main page."
                },
                "options": {
                    "type": "object",
                    "description": "Set options for the scenario.",
                    "properties": {
```

[8] A tale scopo si fa utilizzo della seguente espressione regolare: "^*[a-zA-Z0-9_]+$*"

```
                    "makeTabs": {
                    "type": "boolean",
                    "description": "Show services table into a tabs structure."
                    }
                }
            },
            "services": {
                "type": "array",
                "description": "The list of scenarios' services.",
                "items": { "$ref": "#/$defs/service" },
                "uniqueItems" : true
            }
        }
    },
    "service": {
        "type": "object",
        "required": [ "ID", "Name" ],
        "properties": {
            "ID": {
                "type": "string",
                "pattern": "^[a-zA-Z0-9_]+$",
                "description": "The identificative name of the service."
            },
            "Name": {
                "type": "string",
                "description": "The name of the service."
            },
            "Description": {
                "type": "string",
                "description": "The description of the service."
            },
            "URL": {
                "type": "string",
                "description": "The public URL of the service."
            },
            "dataType": {
                "type": "string",
                  "description": "The type of the provided data (default JSON). Data
can be separated values text."
            },
            "textSeparator": {
                "type": "string",
                 "description": "If data is provided in the form of separated values
text, with this attribute the manager can define the separator char (or text)."
            },
            "table_options": {
                "type": "object",
                "description": "Datatables options format."
            }
        }
```

QUADERNI di GEOFISICA

ISSN 1590-2595 http://istituto.ingv.it/it/le-collane-editoriali-ingv/quaderni-di-geofisica.html/

I QUADERNI DI GEOFISICA (QUAD. GEOFIS.) accolgono lavori, sia in italiano che in inglese, che diano particolare risalto alla pubblicazione di dati, misure, osservazioni e loro elaborazioni anche preliminari che necessitano di rapida diffusione nella comunità scientifica nazionale ed internazionale. Per questo scopo la pubblicazione on-line è particolarmente utile e fornisce accesso immediato a tutti i possibili utenti. Un Editorial Board multidisciplinare ed un accurato processo di peer-review garantiscono i requisiti di qualità per la pubblicazione dei contributi. I QUADERNI DI GEOFISICA sono presenti in "Emerging Sources Citation Index" di Clarivate Analytics, e in "Open Access Journals" di Scopus.

QUADERNI DI GEOFISICA (QUAD. GEOFIS.) welcome contributions, in Italian and/or in English, with special emphasis on preliminary elaborations of data, measures, and observations that need rapid and widespread diffusion in the scientific community. The on-line publication is particularly useful for this purpose, and a multidisciplinary Editorial Board with an accurate peer-review process provides the quality standard for the publication of the manuscripts. QUADERNI DI GEOFISICA are present in "Emerging Sources Citation Index" of Clarivate Analytics, and in "Open Access Journals" of Scopus.

RAPPORTI TECNICI INGV

ISSN 2039-7941 http://istituto.ingv.it/it/le-collane-editoriali-ingv/rapporti-tecnici-ingv.html/

I RAPPORTI TECNICI INGV (RAPP. TEC. INGV) pubblicano contributi, sia in italiano che in inglese, di tipo tecnologico come manuali, software, applicazioni ed innovazioni di strumentazioni, tecniche di raccolta dati di rilevante interesse tecnico-scientifico. I RAPPORTI TECNICI INGV sono pubblicati esclusivamente on-line per garantire agli autori rapidità di diffusione e agli utenti accesso immediato ai dati pubblicati. Un Editorial Board multidisciplinare ed un accurato processo di peer-review garantiscono i requisiti di qualità per la pubblicazione dei contributi.

RAPPORTI TECNICI INGV (RAPP. TEC. INGV) publish technological contributions (in Italian and/or in English) such as manuals, software, applications and implementations of instruments, and techniques of data collection. RAPPORTI TECNICI INGV are published online to guarantee celerity of diffusion and a prompt access to published data. A multidisciplinary Editorial Board and an accurate peer-review process provide the quality standard for the publication of the contributions.

MISCELLANEA INGV

ISSN 2039-6651 http://istituto.ingv.it/it/le-collane-editoriali-ingv/miscellanea-ingv.html

MISCELLANEA INGV (MISC. INGV) favorisce la pubblicazione di contributi scientifici riguardanti le attività svolte dall'INGV. In particolare, MISCELLANEA INGV raccoglie reports di progetti scientifici, proceedings di convegni, manuali, monografie di rilevante interesse, raccolte di articoli, ecc. La pubblicazione è esclusivamente on-line, completamente gratuita e garantisce tempi rapidi e grande diffusione sul web. L'Editorial Board INGV, grazie al suo carattere multidisciplinare, assicura i requisiti di qualità per la pubblicazione dei contributi sottomessi.

MISCELLANEA INGV (MISC. INGV) favours the publication of scientific contributions regarding the main activities carried out at INGV. In particular, MISCELLANEA INGV gathers reports of scientific projects, proceedings of meetings, manuals, relevant monographs, collections of articles etc. The journal is published online to guarantee celerity of diffusion on the internet. A multidisciplinary Editorial Board and an accurate peer-review process provide the quality standard for the publication of the contributions.

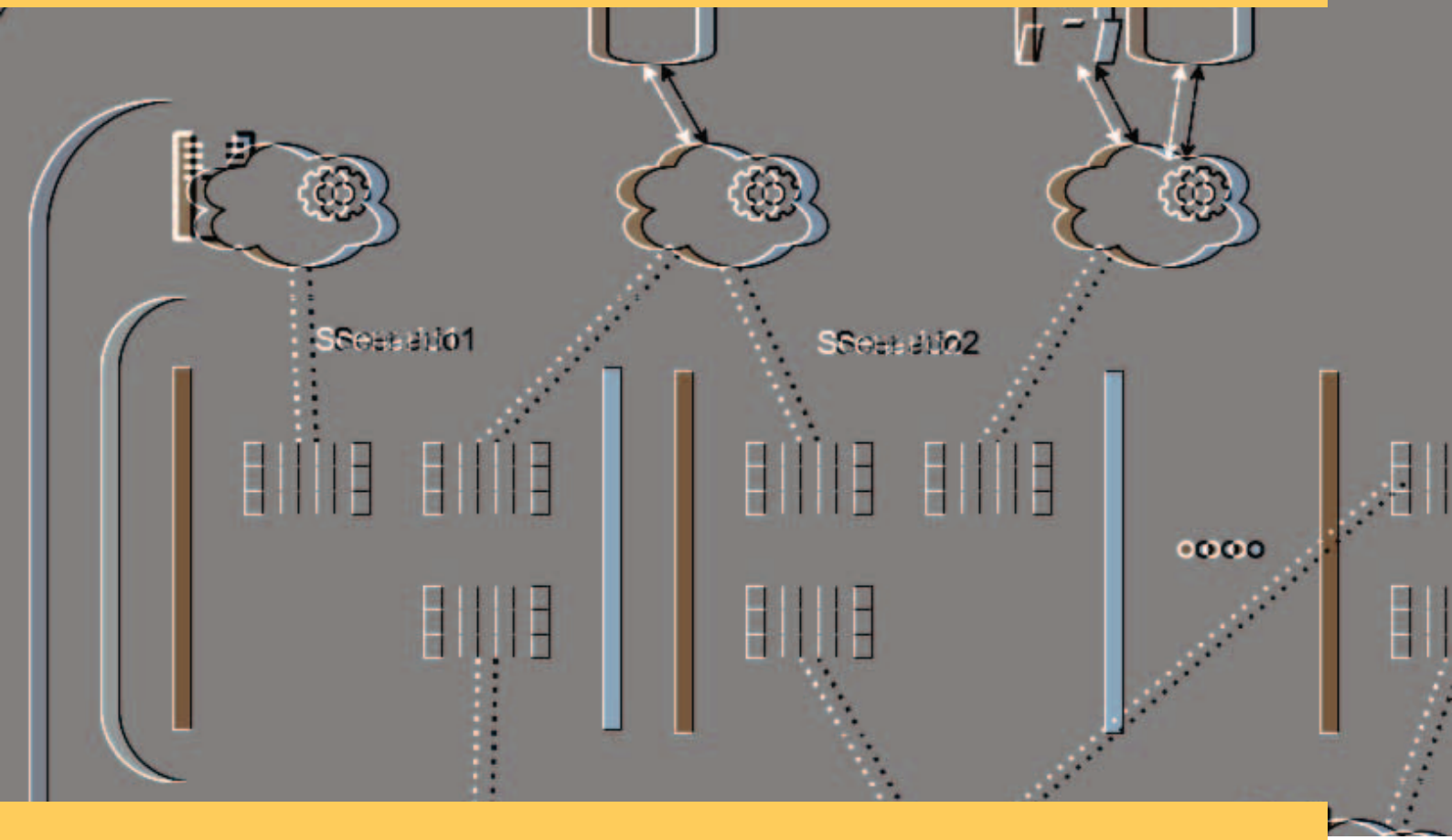
Coordinamento editoriale
Francesca DI STEFANO
Istituto Nazionale di Geofisica e Vulcanologia

Progetto grafico
Barbara ANGIONI
Istituto Nazionale di Geofisica e Vulcanologia

Impaginazione
Barbara ANGIONI
Patrizia PANTANI
Massimiliano CASCONE
Istituto Nazionale di Geofisica e Vulcanologia

Istituto Nazionale di Geofisica e Vulcanologia
Via di Vigna Murata, 605
00143 Roma
tel. +39 06518601

www.ingv.it



ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA