

Form Manager 3

Integrazione con applicativi esterni

Riferimenti	Integrazione con applicativi esterni
Versione SW	Form Manager 3
Autore/i	Cristina Altomare
Data	07/09/2018
Versione	1.5
Ambiente di test VPN	Server FM3 e tokenissue: devsrv03.erossi.org:8080 Collegarsi in VPN al server devsrv03.erossi.org su cliente GPI S.P.A.

SOMMARIO

ACRONIMI E GUIDE UTILI.....	3
1. FORM MANAGER 3.....	4
Requisiti di installazione.....	4
Componenti di Form Manager 3.....	4
Configuratore delle form.....	5
Campi della form.....	6
Operazioni sulla form.....	8
Preinizializzazione della form.....	10
Dati temporanei della form.....	10
Dati della form da salvare.....	10
Libreria HTML.....	11
Libreria Javascript.....	11
Gestione allegati (immagini, pdf...).....	12
Cambio lingua all'interno della form	13
Configuratore dei report di stampa.....	15
Configurazione applicativi chiamanti.....	17
Interrogazione di servizi esterni nella form.....	18
Interrogazione di viste su database esterni.....	19
Interrogazione di cataloghi definiti su NGH.....	19
2. PREMessa PER L'UTILIZZO DEI SERVIZI DI INTEGRAZIONE.....	20
3. WS PER OTTENERE IL TOKEN DI AUTENTICAZIONE.....	20
4. WS PER CREARE O AGGIORNARE UN'ISTANZA DI UNA FORM.....	21
5. WS PER RECUPERARE UNA FORM A PARTIRE DALL'ID DELL'ISTANZA.....	26
6. WS PER CANCELLARE UN'ISTANZA DATO L'ID.....	27
7. WS PER RECUPERARE IL PDF DI STAMPA DELLA FORM.....	29
8. WS PER PUBBLICARE UN'ISTANZA DATO L'ID.....	31
9. RUNTIME.....	32
10. POSTMESSAGE.....	36
11. CONCETTI.....	39
Configuratore dei concetti.....	39
Utilizzo dei concetti nella form.....	40
Definizione dei concetti tramite funzione Javascript.....	41
Valorizzazione dei concetti nei servizi chiamati da applicativi esterni.....	41
Recupero dei concetti nella form.....	41
12. WS PER RECUPERARE VALORI DEI CONCETTI E ID DELLE ISTANZE TRAMITE CONCETTO.....	42
13. WS PER RECUPERARE LE ISTANZE TRAMITE CONCETTO.....	45
14. WS PER RECUPERARE VALORI DEI CONCETTI E ID DELLE ISTANZE TRAMITE PATIENT ID.....	49
15. WS PER RECUPERARE ULTIMI VALORI DEI CONCETTI RILEVATI.....	51
16. WS PER RECUPERARE LISTA DEI CONCETTI DEFINITI SU INSTALLAZIONE.....	54
17. WS PER RECUPERARE LISTA DEI CONCETTI ESPOSTI DA UN TEMPLATE.....	55
18. SUMMARY FIELDS.....	57
Utilizzo dei summary nella form.....	57
Recupero dei summary nei servizi chiamati da applicativi esterni.....	58
19. WS PER RECUPERARE I SUMMARY FIELDS TRAMITE ISTANZA.....	59
20. INTEGRAZIONE CON REPOSITORY.....	61
Parametri di configurazione di FORMMANAGER.....	61
Parametri di configurazione di IEREPOSITORY.....	61
Parametri di configurazione della singola form.....	62
Metadati.....	64

ACRONIMI E GUIDE UTILI

In premessa alla documentazione che segue, introduciamo alcuni acronimi che saranno utilizzati in questo documento e alcune guide utili di approfondimento.

- **FORM:** Interfaccia utente complessa per la raccolta di dati. Nel documento è utilizzato anche il sinonimo **templa-te** per indicare la form.
- **ISTANZA:** Template valorizzato a runtime
- **FORM MANAGER:** Software GPI che permette una rapida realizzazione di FORM attraverso l'uso di un'interfaccia grafica
- **NGH:** Software GPI per la gestione della cartella clinica ambulatoriale e di ricovero
- **HTML:** Linguaggio di markup per la realizzazione di interfacce web
(Guida HTML5: <https://www.w3schools.com/html/default.asp>)
- **CSS:** Linguaggio per definire lo stile di un documento HTML
(Guida CSS: <https://www.w3schools.com/Css/>)
- **Javascript:** Linguaggio di programmazione orientato al web
(Guida JavaScript: <https://www.w3schools.com/js/default.asp>)
Funzioni di calcolo matematico: JavaScript Math object https://www.w3schools.com/js/js_math.asp
- **Bootstrap:** Framework CSS per la realizzazione di interfacce responsive
(Guida Bootstrap: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>, Tema Flatify: <http://wrapbootstrap.com/preview/WB0977873>)
- **Angular:** Framework Javascript per lo sviluppo di applicazioni web client-side
(Guida AngularJS - direttive: <https://docs.angularjs.org/api/ng/directive> , Integrazione Bootstrap – Angular JS <https://angular-ui.github.io/bootstrap/>)

1. FORM MANAGER 3

Requisiti di installazione

Per utilizzare il FormManager 3 occorre installare

- **WildFly 10.1**
- uno **schema db HIS** Oracle o Postgres per ospitare l'interfaccia di configurazione (database base di NGH)
- l'ultima versione di **NGH** (GPI)
- **Mongodb 3.6+**
- **TokenIssueService** (GPI)

Su NGH far configurare le seguenti utenze (già configurate per l'ambiente di test devsrv03) di integrazione:

- user: **fm3**, password: **T3stFM3!** (le stesse credenziali utilizzate nel tokenissue.properties), **status = FM3**
- user: **utentefm.terapie** (o **utentefm.opera** o **utentefm.silor** a seconda dell'applicazione esterna da integrare), con password qualsiasi e **status = FM3**

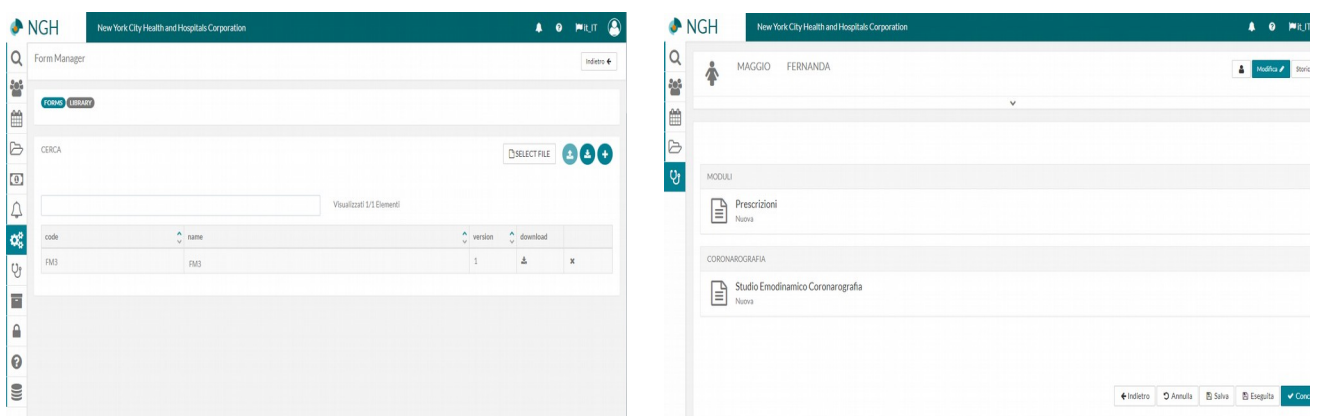
Su NGH far aggiungere un'utenza per poter configurare le form dal menu Form Manager (definita come sotto). A questa utenza associare il profilo condiviso FM3 Configuratore (già presente nell'ambiente di test devsrv03). Accedere quindi all'url <http://devsrv03.erossi.org:8080/HIS> e utilizzare le seguenti credenziali che permettono **solo l'accesso al Form Manager**:

- nome utente: **config.fm3**
- password: **aaa**

Componenti di Form Manager 3

Form Manager 3 si compone di

- un **configuratore**: modulo che permette di disegnare i template da istanziare e valorizzare
- un **runtime**: componente che permette di istanziare e valorizzare un template creato nel configuratore



Configuratore delle form

Cliccare sul menu **Form Manager** indicato dal seguente tab .

La pagina presenta la lista delle form configurate:

FORMSHTMLLIBRARYJSLIBRARYAPPLICAZIONI

CERCA

Mostra cancellatiSELECT FILE

Visualizzati 5/87 Elementi

Codice	Nome	Versione	Stato	Scarica	
test_fm3	Test con report	2	PUBLISHED		
assessment-nanda-1	1 - Health promotion	1	DRAFT		
NANDA_DOMAIN_1	1 - Health promotion	1	PUBLISHED		
NANDA_DOMAIN_2	2 - Nutrizione	1	PUBLISHED		
NANDA_DOMAIN_3	3 - Eliminazione	1	PUBLISHED		

Mostra5

Prima1234Ultima

- Sulla lista sono presenti le informazioni sulla **versione** e sullo **stato** del template
- Lo **stato** di un template può essere
 - DRAFT** (in bozza): il template può essere modificato senza creare nuove versioni. Il template non può essere istanziato
 - PUBLISHED** (pubblicato): il template può essere istanziato. Nuove modifiche al template implicheranno la creazione di una nuova versione in stato DRAFT
 - INVALID** (riga barrata): il template risulta non valido (quindi non può essere istanziato) perché è stato cancellato o è stata creata una nuova versione
- E' possibile **ordinare** la lista in funzione dei campi cliccando sulle frecce apposite
- Per poter **creare** una nuova form è possibile cliccare sul bottone di **Aggiungi** e crearne una da zero oppure utilizzare il **SELECT FILE** per caricare un file json di una form preesistente.



Se il file json caricato è relativo a una form con un report associato non presente nell'installazione corrente comparirà in fase di caricamento il seguente popup e nella form caricata il report non risulterà configurato.

Informazione

Non è stato trovato il file per la stampa. Il salvataggio continua comunque

OK

- La lista è prefiltrata mostrando solo le **form valide**. Cliccando sul bottone **Mostra cancellate** è possibile visualizzare anche le versioni invalidate delle form

- Una form può essere invalidata utilizzando il **Rimuovi** (ultima colonna). Una form invalidata non potrà più essere istanziata.
- Ogni form può essere scaricata tramite il **download** sulla singola form, oppure è possibile scaricare tutte le form utilizzando il bottone di download in alto a destra.

Cliccando sulla **singola form** si accede al **configuratore** la cui interfaccia è la seguente:

The screenshot shows the Form Manager configuration interface. At the top, there are three input fields: 'Codice' (test_fm3), 'Nome' (Test con report), and 'Versione' (2). To the right of these fields are buttons: 'Annulla', 'Salva', 'Opzioni template', 'Pubblica', 'Scarica', and 'Anteprima'. Below these fields is a large area divided into two parts. The left part is the 'Editor HTML' with a code editor showing HTML code for a form. The right part is the 'Mostra in tempo reale' (Live preview) showing the form's appearance. The form has a title 'TEST FORM CON REPORT 2' and a field 'Field1 - mod' with a label 'Enter a field value'. Below the field is a label 'The value is:' followed by a text input field. At the bottom of the live preview, there are checkboxes for 'Mostra JSON viewData', 'Mostra JSON formData', and 'Dati validi'.

Campi della form

Per ogni form è possibile indicare

- **Codice Form:** codice della form, utilizzato nei servizi di integrazione per fare riferimento alla form stessa
- **Nome Form:** nome della form visualizzato a runtime nella chiamata da applicativi esterni
- **Versione:** versione della form visualizzata a runtime nella chiamata da applicativi esterni. Le istanze create con le versioni precedenti rimarranno associate alle versioni con cui sono state create

The screenshot shows the Form Manager configuration interface with three input fields: 'Codice' (test_fm3), 'Nome' (Test con report), and 'Versione' (2).

Per disegnare una form vi sono a disposizione i seguenti editor:

- **Editor HTML:** sezione dove definire i tag HTML. Per la definizione e il tutorial sull'HTML far riferimento alla sezione **Acronimi** della presente documentazione.

L'editor segnala automaticamente se ci sono *errori* nel codice HTML scritto. In alto segnala il numero di errori presenti, in corrispondenza della riga dove l'errore è presente, segnala il tipo di errore (tramite tooltip, passando il puntatore del mouse sopra l'icona in rosso della riga, come evidenziato in figura).

The screenshot shows the HTML editor with a red error icon in the top right corner. A tooltip is displayed over the error icon, showing the error message: 'Special characters must be escaped : [< >].' and 'Tag must be paired, no start tag: [</label>]'. The code editor shows HTML code for a form with a label 'Valutazione' and a text input field.



- L'editor mette a disposizione 3 icone:
 - *Libreria HTML*: bottone che permette di navigare la libreria delle componenti HTML già realizzate (per maggiori dettagli vedi la sezione **Libreria HTML** della documentazione)
 - *Formatta codice*: cliccando su questo bottone il codice sarà automaticamente formattato
 - *Cerca/sostituisci*: permette di ricercare e/o sostituire testi all'interno dell'editor
- **Editor CSS**: sezione dove definire lo stile degli elementi HTML tramite il CSS. Per la definizione e il tutorial sul CSS far riferimento alla sezione **Acronimi** della presente documentazione.

L'editor segnala automaticamente se ci sono *errori* nel codice CSS

scritto analogamente all'editor HTML.



L'editor mette a disposizione 2 icone:

- *Formatta codice*: cliccando su questo bottone il codice sarà automaticamente formattato
- *Cerca/sostituisci*: permette di ricercare e/o sostituire testi all'interno dell'editor

Editor CSS  

```
1 .radio-form{
2   padding:5px
3
4 }
5 .panel-form{
6   border: groove ;
7   border-color: #92a8d1;
8 }
9
```


- **Editor Javascript**: sezione dove definire il codice Javascript. Per la definizione e il tutorial sul Javascript far riferimento alla sezione **Acronimi** della presente documentazione. L'editor segnala automaticamente se ci sono *errori* nel codice Javascript scritto analogamente all'editor HTML. L'editor mette a disposizione 2 icone:
 - *Libreria Javascript*: bottone che permette di navigare la libreria delle funzioni Javascript già realizzate (per maggiori dettagli vedi la sezione **Libreria Javascript** della documentazione)
 - *Formatta codice*: cliccando su questo bottone il codice sarà automaticamente formattato
 - *Cerca/sostituisci*: permette di ricercare e/o sostituire testi all'interno dell'editor

Editor Javascript  

```
1 var testFunction = function() {
2   console.dir(formData);
3   console.dir(initData);
4   if (initData && initData.patient) {
5     if (!formData.patient) {
6       formData.patient = {};
7     }
8     formData.patient.name = initData.patient.name;
9     formData.patient.surname = initData.patient.surname;
10  }
11  };
12
13 testFunction();
```

La pagina di modifica di una form presenta infine una sezione dedicata all'**anteprima** della form stessa. Ogni volta che si effettua una modifica in uno degli editor, l'anteprima è ricaricata. Per disabilitare il reload automatico dell'anteprima è sufficiente cliccare su **Mostra in tempo reale**: comparirà un bottone di **Ricarica preview**, cliccando sul quale l'anteprima sarà ricaricata su richiesta dell'utente.

Mostra in tempo reale ☐

 Ricarica preview

Operazioni sulla form

Le operazioni eseguibili sulla form sono le seguenti:



- **Annulla:** torna alla lista delle form senza salvare le modifiche
- **Salva:** salva la form
 - non è possibile modificare una form di una versione precedente all'ultima presente (che sia in stato PUBLISHED o DRAFT). Pertanto il bottone Salva risulterà **abilitato** solo sull'**ultima versione** del template creato. Il bottone Salva inoltre sarà **disabilitato** se ci sono **errori** in uno degli editor.
- **Pubblica:** pubblica la form
 - solo una form pubblicata può essere istanziata
 - modificando l'ultima form già pubblicata e cliccando su Salva, in automatico viene creata una nuova versione della stessa (in stato=DRAFT) e invalidata la precedente. Le form create con le vecchie versioni rimarranno associate alle vecchie versioni.
 - Non è possibile pubblicare una form già pubblicata (il bottone Pubblica risulterà disabilitato)
- **Scarica:** effettua il download della form
- **Anteprima:** mostra in un'altra pagina l'anteprima della form a tutto schermo
- **Editor Bundles:** cliccando sul bottone si apre una finestra con i bundles associati alla form, filtrabili per lingua tramite il menu a tendina **Lingua**

Opzioni template - test_integrazione_03 - Test Integrazione 03 - v.10



Lingua

Filtra ...

Visualizzati 8/8 Elementi

Lingua	Chiave	Valore	
it_IT	test	test	✕
it_IT	test01	test01	✕
en_GB	test	test	✕
en_GB	test01	test01	✕
es_ES	test	test	✕

Mostra

5

Prima

«

1

2

»

Ultima

E' possibile **eliminare** un bundles cliccando sulla x posta sulla riga corrispondente. E' possibile **ricercare** un bundle tramite il campo 'Filtra' che risulta visibile solo in caso di almeno 5 bundles configurati.

E' possibile **aggiungere** un nuovo bundle cliccando sul bottone verde di aggiunta posto in alto a destra. Il bottone risulterà visibile solo per i template in stato Bozza (cioè modificabili). Cliccando su Aggiungi comparirà la sezione seguente dove definire la chiave del bundle e i possibili valori per ogni lingua. Non è necessario valorizzare tutte le lingue (in tal caso il bundle sarà vuoto per la lingua corrispondente).

Chiave

Valore per lingua it_IT

Valore per lingua en_GB

Valore per lingua es_ES

Valore per lingua de_DE

Annulla

Salva

- **Opzioni template:** cliccando sul bottone si apre una finestra con le opzioni associabili alla form
La modifica e il salvataggio di queste opzioni sono contestuali al click sul bottone Salva della finestra e non impli-
cheranno la creazione di una nuova versione della form.

Le opzioni relative al repository, e meglio dettagliate nel capitolo **Integrazione con il repository**, sono:

Invio al Repository abilitato



CDA2 abilitato



Invia txt file



Annullamento abilitato



Firma digitale abilitata



- **Invio al Repository:** se abilitato, la form potrà essere inviata al repository (oggetto patient obbligatorio)
- **Firma digitale abilitata:** se abilitata, la form potrà essere firmata digitalmente (oggetto patient obbligatorio)
- **Annullamento abilitato:** se abilitato sarà consentito l'annullamento del documento sul repository
- **CDA2 abilitato:** se abilitato, nell'invio al repository al documento sarà allegato il CDA2 definito nella sezione xml che si visualizza quando abilito l'opzione CDA2
- **Invia txt file:** se abilitato, sarà consentito l'invio di un file txt insieme al documento da inviare al repository

Le **opzioni sui report:** le seguenti opzioni riguardano i **report di stampa** associati alla form

Nome Report

Anteprima di Stampa

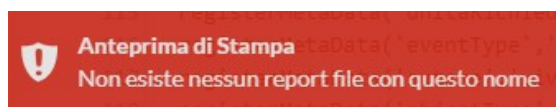
Nome Report per applicazione

Applicazione

Salva

Nome Report per applicazione		
test_cristina_01	MI_Scheda_Richiesta.jasper	✕
test_cristina_02	test_fm3.jasper	✕

- **Nome report:** nome del file .jasper (es: *test_fm3.jasper*) da associare alla form. Inserendo almeno 3 caratteri apparirà una lista di *suggerimenti* tra i report già presenti. Utilizzando il bottone **Anteprima di stampa** si può visualizzare il pdf del report solo se il *report esiste* ed è *correttamente configurato*. Questo report sarà quello utilizzato come default nel caso non siano configurati report specifici per applicazioni chiamante (vedi punto seguente). In caso di file non caricato sul server, cliccando su Anteprima si otterrà il seguente errore



- **Nome report per applicazione e applicazione:** è possibile, per la stessa form, configurare report diversi per diversi applicativi chiamanti diversi. Un esempio pratico di utilizzo è quando una form è utilizzata da più clienti le cui stampe differiscono solo per intestazione e/o logo. In questo caso è possibile associare a **ogni applicazione chiamante un report diverso**, come mostrato in figura. A **runtime** e in chiamata al servizio **printExt**, sarà restituita la stampa associata all'applicativo chiamante corrispondente (in caso di non configurazione specifica del report per applicativo chiamante, sarà utilizzata la stampa di **default** definita in Nome report). In caso di file non presente tra i report inseriti in Form Manager sarà restituito il seguente errore e l'associazione report-applicazione chiamante non sarà salvata:

Informazione
Non è stato trovato il file per la stampa. Il salvataggio è annullato.
OK

Nota sui report: i report vanno configurati da interfaccia (vedi sezione **Configuratore report di stampa** del seguente documento) oppure caricati manualmente sul server, nella cartella di wildfly **standalone/configuration/gpi/his/reports/fm3**. Per referenziare i campi nel report, se il campo è del tipo *formData.field1* utilizzare la sintassi *formDocument.formData.field1*. In caso alla form sia associato un **report non configurato**, al salvataggio sarà fornito il seguente warning: **Informazione: Non è stato trovato il file per la stampa. Il salvataggio continua comunque** e non sarà salvato nessun report di default associato alla form.

Preinizializzazione della form

E' possibile preinizializzare qualsiasi campo definito per la form utilizzando l'oggetto **initData** passato nella chiamata al Form Manager. E' possibile referenziare direttamente il campo utilizzando la direttiva `ng.model="initData.field1"`

```
<input type="text" ng-model="initData.patient.name" disabled>
```

Oppure, se voglio salvare il valore del campo passato nella form, nel javascript scriverò **formData.field1 = initData.field1**. Data la seguente form con `ng-model="formData.patient.name"` per Patient Name e `ng-model="formData.patient.surname"` per Patient Surname, se voglio istanziarli e poi **salvarli** scriverò una **funzione javascript** del tipo mostrato in figura.

FORM DI TEST - VERSIONE 1

Patient name
Alessandra

Patient Surname
Illuminati

Notes

```

1 var testFunction = function() {
2   console.dir(formData);
3   console.dir(initData);
4   if (initData && initData.patient) {
5     if (!formData.patient) {
6       formData.patient = {};
7     }
8     formData.patient.name = initData.patient.name;
9     formData.patient.surname = initData.patient.surname;
10  }
11 };

```

Se nella chiamata ai servizi esterni, si specifica l'oggetto **patient** tra i **parametri di input** del **body** (vedi appositi capitoli successivi), è possibile inizializzare i dati relativi al paziente utilizzando l'oggetto **fmPatient** nel seguente modo: **ng-model="fmPatient.firstName"**. La definizione dell'oggetto patient è riportata a pagina 20, un esempio a pagina 26.

Dati temporanei della form

E' possibile definire dati temporanei nel modello dei dati, utilizzati solo all'interno della form e non salvati, tramite l'oggetto **viewData**. Per controllare i viewData definiti, cliccare su **Mostra JSON viewData** nella parte sinistra della pagina.

Javascript

```
viewData.selectSegments=[{code:'CDXI', side:'dx'},
{code:'CDXII', side:'dx'},
{code:'CDXIII', side:'dx'},
```

Mostra JSON viewData

```
{ "selectSegments": [ { "code": "CDXI", "side": "dx", "$$hashKey": "object:16" }, { "code": "CDXII", "side": "dx", "$$hashKey": "object:17" }, { "code": "CDXIII", "side": "dx", "$$hashKey": "object:18" } ] }
```

Dati della form da salvare

Per definire quali valori salvare della form, occorre utilizzare l'oggetto **formData**. Per controllare i formData definiti, e i rispettivi valori, cliccare su **Mostra JSON formData** nella parte sinistra della pagina.

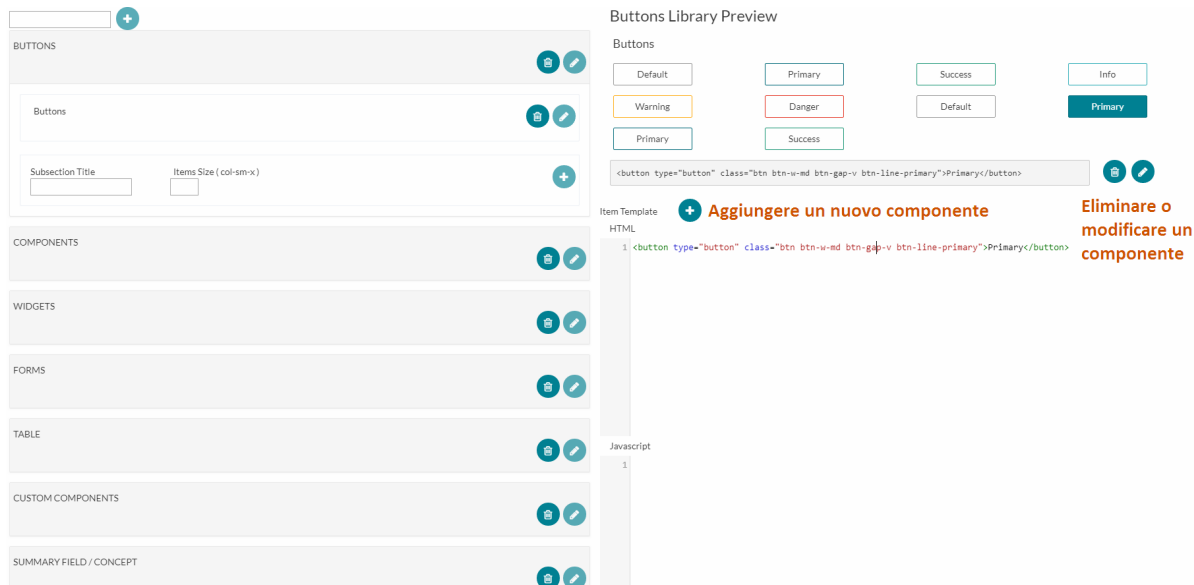
```
<label>Anamnesi familiare</label>
<textarea rows="4" class="form-control" required ng-model="formData.anamnesiFamiliare">
```

Mostra JSON formData

```
{ "anamnesiFamiliare": "note di anamnesi", "sdo": "11111111" }
```

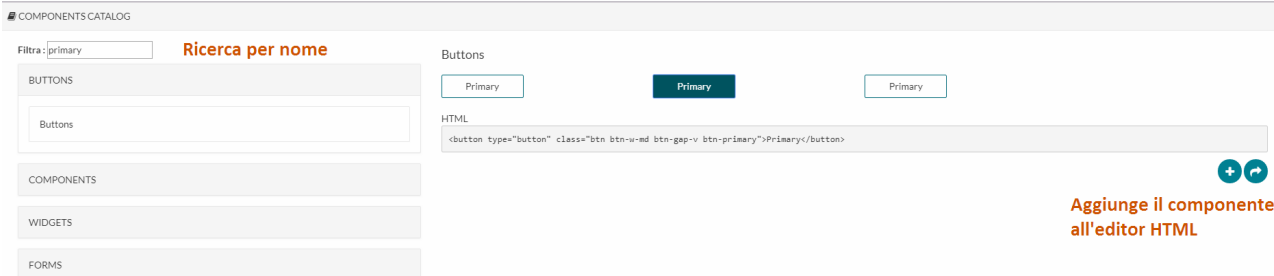
Libreria HTML

In questa sezione è possibile consultare ma anche definire nuove componenti HTML utili per la realizzazione della form.



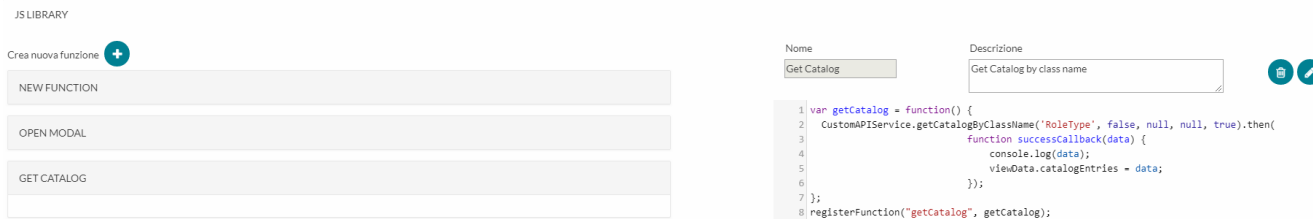
Per utilizzare un componente è possibile consultare la libreria direttamente dal bottone apposito dell'editor HTML, attraverso il quale è possibile ricercare per nome direttamente il componente di interesse e importarlo nel proprio editor, cliccando sui bottoni in basso a destra (come mostrato in figura sotto).

Seleziona componente HTML da aggiungere



Libreria Javascript

In questa sezione è possibile consultare ma anche definire nuove funzioni JavaScript utili per la realizzazione della form.



Per utilizzare una funzione già presente in libreria è possibile consultare la libreria direttamente dal bottone apposito dell'editor Javascript, attraverso il quale è possibile ricercare per nome direttamente la funzione di interesse e importarla nel proprio editor, analogamente a quanto illustrato per la libreria HTML.

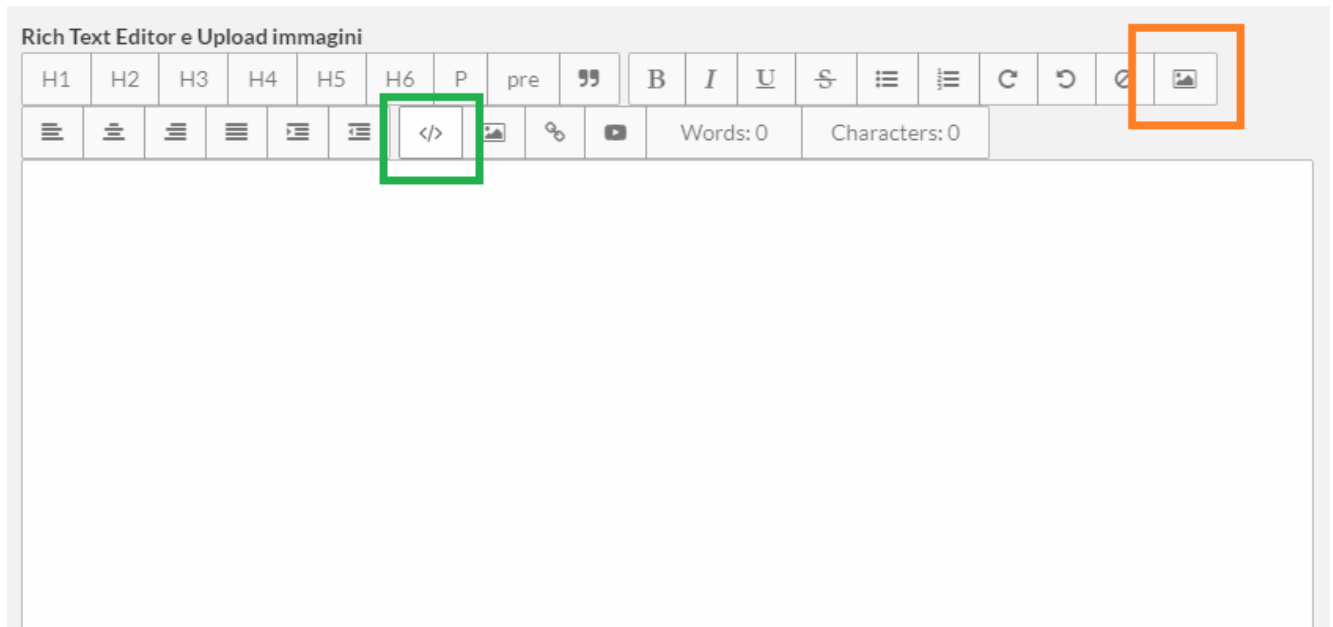
Citiamo le funzioni **preSaveFunction**, **postSaveFunction** e **prePublishFunction** che consentono, rispettivamente, di eseguire delle operazioni prima e dopo il salvataggio dell'istanza e prima della pubblicazione dell'istanza.

Gestione allegati (immagini, pdf...)

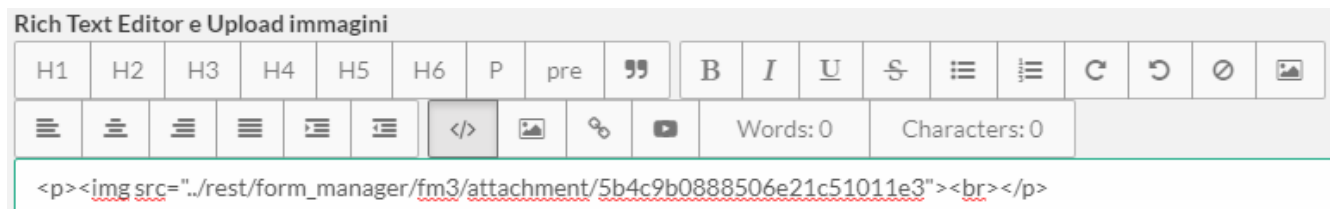
Per caricare un **immagine** o un **documento in pdf** si può utilizzare il componente di Gestione allegati inserito nella **libreria**

html: `<text-angular ng-model="formData.referto"></text-angular>`

Il codice sopra corrisponde all'elemento di **Rich Text Editor** sotto mostrato:



Cliccando sul bottone evidenziato in arancione in figura è possibile **caricare** in Form Manager 3 un **documento** o un **immagine**. Una volta caricata l'immagine o il pdf, cliccando sul bottone in verde sarà visibile il **percorso relativo** al file caricato, del tipo: `../rest/form_manager/fm3/attachment/5b4c9b0888506e21c51011e3`



Sarà quindi possibile utilizzare questo percorso per mostrare l'**immagine** nella form come nell'esempio sotto

``

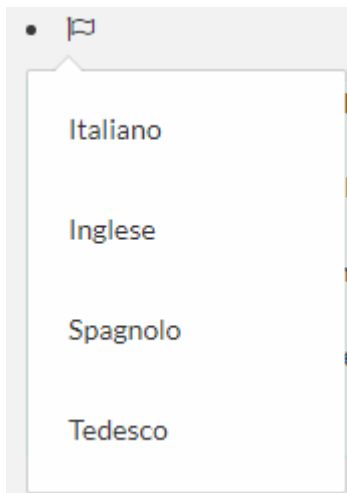
oppure **aprire il documento pdf** inserendo il seguente codice nell'editor javascript:

```
function testOpenPdf() {  
    window.open('../rest/form_manager/fm3/attachment/5b4c9b0888506e21c51011e3');  
}
```

Nell'ambiente di test devsrv03 è riportata una form di esempio con codice **test_allegati** di utilizzo del componente per l'apertura del PDF. Cliccando sul bottone Open PDF si apre una seconda finestra con il PDF caricato.

Cambio lingua all'interno della form

Nella libreria HTML, in Custom Components è presente il componente Language. Il componente è un bottone con menu a tendina che consente di **cambiare il linguaggio utilizzato all'interno della form**:



Renderizzato dal seguente componente definito nell'Editor HTML.

```
<li class="dropdown open" uib-dropdown="" auto-close="outsideClick" ><a class="dropdown-toggle"
uib-dropdown-toggle="" aria-haspopup="true" aria-disabled="false" > <i class="fa fa-flag-o"></i>
</a>
<ul class="dropdown-menu with-arrow list-langs" role="menu">
<li ng-repeat="language in viewData.languages" style=""><a href="javascript:;"
data-ng-click="changeLanguage(language.code)"><span class="ng-binding">{{language.description}}</span></a>
</li>
</ul>
</li>
```

Le lingue che popolano il menu a tendina sono settate tramite funzione Javascript. Nell'esempio sotto riportato la lista delle lingue è presa dal catalogo Language di NGH:

Editor Javascript

```
1 viewData.selectLanguage=false;
2 CustomAPIService.getCatalogByClassName('Language',true,null,null,true).then(
3   function successCallback(data) {
4     viewData.languages = data;
5   });
```

Per definire più traduzioni per uno stesso campo, occorre utilizzare l'Editor dei Bundles

Editor Bundles

Lingua

Lingua	Chiave	Valore	
it_IT	test	Prova	✕
en_GB	test	English Test	✕

E nell'Editor HTML definire le voci in questo modo come nell'esempio `<label>{{ 'test' | translate}}</label>`



Configuratore dei report di stampa

Accedendo al FormManager da NGH, per configurare i report di stampa (cioè i file **.jasper**) da associare alle form, occorre cliccare sul menu **REPORT**:

FORMS **HTML LIBRARY** **JS LIBRARY** **APPLICAZIONI** **REPORT** **CONCETTI**





Si aprirà una pagina con una lista dei report già censiti.

Report


Upload file jasper per report ☒ Upload file immagini per report ☐ Upload file bundle per report ☐

Filtra ... Visualizzati 4/4 Elementi

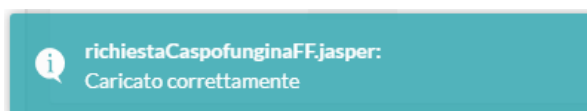
Report	Scarica	
MI_Scheda_Richiesta.jasper		×
bah.jasper		×
esame-obiettivo-neurochirurgia.jasper		×
test_fm3.jasper		×

Per caricare un nuovo report di stampa, selezionare la voce **Upload file jasper per report** e cliccare sul bottone **Select file**. Si aprirà il File Manager per scegliere il report da caricare. Una volta selezionato comparirà il nome del file caricato e l'opzione **Sovrascrivere le versioni precedenti** da selezionare nel caso il report esista già e occorra sovrascriverlo. Cliccando sul **bottone di upload** il file sarà caricato sull'installazione di NGH.

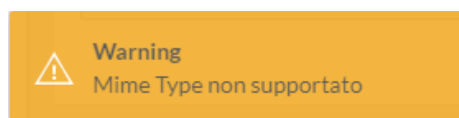
Report

richiestaCaspofunginaFF.jasper **Sovrascrivere le versioni precedenti** ☐ 

Se il caricamento è andato a buon fine comparirà il seguente messaggio di conferma in fondo alla pagina:



Sarà possibile caricare solo file con estensione **.jasper**. Se si prova a caricare un file con altra estensione il salvataggio sarà interrotto e comparirà il seguente messaggio:



Ogni file della lista potrà essere **rimosso** o **scaricato** in locale tramite le funzioni disponibili su ogni riga.

E' possibile caricare anche le **immagini** da associare ai report selezionando la voce **Upload file immagini per report**. Selezionando questa voce comparirà la lista delle immagini già caricate e sarà possibile caricare una nuova immagine.

Upload file immagini per report



Le estensioni per ora supportate sono **.png, .jpg, .jpeg, .gif**. Le immagini saranno caricate nella cartella **images** del server, da referenziare nell'apposito report come nel seguente esempio: `<imageExpression>![CDATA["images/logo.jpg"]]></imageExpression>`.

E' possibile caricare anche i **bundles** da associare ai report selezionando la voce **Upload file bundle per report**. Selezionando questa voce comparirà la lista delle file bundle già caricati e sarà possibile caricare un nuovo file.

Upload file bundle per report



L'unica estensione supportata per questo tipo di file è **.properties**. Questi file saranno caricati nella cartella **bundles** e referenziati nel report come nell'esempio precedente sulle immagini.

In fase di associazione del report alla form, in **Opzioni template**, digitando almeno 3 caratteri saranno filtrati i report disponibili che iniziano con i 3 caratteri digitati come nell'esempio:

Nome Report



Anteprima di Stampa

test_fm3.jasper

zione

Applicazione


Cliccando su **Anteprima di Stampa**, se l'anteprima va a buon fine, vorrà dire che il report è stato caricato correttamente e che non ci sono errori sul report stesso.

Configurazione applicativi chiamanti










Accedendo al FormManager da NGH, per configurare gli applicativi chiamanti occorre cliccare sul menu **APPLICAZIONI**:



Si aprirà una pagina con una lista delle applicazioni già censite. Queste applicazioni fanno riferimento al parametro **ap-
plication** inserito nell'header del servizio **urlByCodeAndVersionExt**.

Applicazioni 

Visualizzati 12/12 Elementi

Applicazione	
IE-TERAPIEddddddddd	
NGH	 
IE-OPERA	 
IE-TERAPIE	 
IE-SILOR	 

Mostra


Prima < 1 2 3 > Ultima

Per **aggiungere** un applicativo chiamante occorre cliccare sul bottone



Applicazioni

Applicazione

 Aggiungi

 Annulla

Inserire quindi il nome dell'applicazione e cliccare su **Aggiungi**.

Per **eliminare** un applicativo chiamante è sufficiente cliccare sul bottone di elimina, posto nella colonna sinistra. Dopo la cancellazione la riga sarà barrata e l'applicativo risulterà non valido. Cliccando sul bottone di reload posto nella colonna sinistra sarà possibile riabilitare l'applicazione.

Per modificare un applicativo chiamante è sufficiente cliccare sulla riga corrispondente. Cliccando sarà visibile il campo del nome dell'applicazione da modificare. Cliccando sul tasto Salva saranno salvate le modifiche.

Applicazioni

Applicazione

 Salva

 Annulla

Interrogazione di servizi esterni nella form

Nel configurare una form è possibile **recuperare dei valori** utilizzando **chiamate a servizi SOAP e REST**. Queste chiamate vanno configurate nell'editor **Javascript** del configuratore come negli esempi sotto riportati. Nella **libreria JavaScript** è possibile trovare i due esempi di chiamata REST cercando **GET VALUE FROM REST SERVICE**.

Per vedere un esempio nell'**ambiente di test**, fare riferimento alla **form di esempio** creata con code **test_rest_service** e vedere il risultato nella console del browser.

Esempio chiamata servizio REST in GET

```
function testRestServiceGET() {  
  var object = {};  
  object.headers = {};  
  object.headers["Content-type"] = "application/json; charset=UTF-8";  
  CustomAPIService.callRestService("http://jsonplaceholder.typicode.com/posts/1", "GET", object).then(  
    function successCallback(data) {  
      formData.freetext = data.body;  
      formData.prova = data.title;  
      console.dir(data);  
    });  
}  
registerFunction("testRestServiceGET", testRestServiceGET);
```

Esempio chiamata servizio REST in PUT

```
function testRestServicePUT() {  
  var object = {};  
  object.payload = {  
    id: 1,  
    title: 'foo',  
    body: 'bar2',  
    userId: 1  
  };  
  object.headers = {};  
  object.headers["Content-type"] = "application/json; charset=UTF-8";  
  CustomAPIService.callRestService("http://jsonplaceholder.typicode.com/posts/1", "PUT", object).then(  
    function successCallback(data) {  
      console.dir(data);  
    });  
}  
registerFunction("testRestServicePUT", testRestServicePUT);
```

Esempio chiamata servizio SOAP

```
CustomAPIService.callWsService(<ws_url>,<soap_request_xml>,<auth_header>).then(  
  function successCallback(data) {  
    console.dir(data);  
  });
```

Interrogazione di viste su database esterni

Nel configurare una form è possibile **recuperare dei valori** interrogando direttamente delle **viste** su database esterni, create sull'utente **his** del database. L'utente his creato appositamente per FM3 ha i soli **grant di SELECT** per consentire la **sola lettura** da queste viste.

Un esempio è riportato nella **libreria Javascript** (cercare **GET VALUES FROM VIEW**). L'interrogazione va definita nell'editor Javascript come nell'esempio mostrato in figura. Per vedere un esempio nell'**ambiente di test**, fare riferimento alla **form di esempio** creata con code **testQuery**. La funzione definita nell'esempio interroga la vista *cat_agreement* di his e mostra i risultati nella *console* del browser:

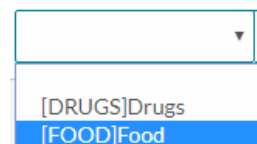
```
function testQuery() {  
    CustomAPIService.executeQuery("SELECT * FROM his.cat_agreement").  
        function successCallback(data) {  
            console.dir(data);  
        };  
}  
registerFunction("testQuery", testQuery);
```

Interrogazione di cataloghi definiti su NGH

E' possibile definire su NGH dei **cataloghi** che possono essere richiamati all'interno della form per popolare delle liste (es: lista comuni, lista diagnosi ecc).

E' possibile utilizzare un componente HTML già presente nella **libreria HTML** (cercare **Catalog Search Input**). Nell'esempio sotto riportato si utilizza il catalogo NGH **AllergyType** definito nell'attributo **class-name** del componente. Il risultato in HTML del codice sopra implementato è una lista con i valori Tipo di Allergia, come mostrato a destra.

```
<catalog-search-input validation-form="validationForm" field-properties="fieldProperties"  
    field-prefix="" field-name="allergyType" class-name="AllergyType"  
    model-field="catalogEntry.allergyTypeCode"  
    code-as-model="true">  
</catalog-search-input>
```



E' possibile in alternativa utilizzare un componente JavaScript già presente nella **libreria Javascript** (cercare **GET CATALOG**). Nell'esempio sotto riportato si utilizza il catalogo NGH **RoleType** come primo parametro della funzione **getCatalogByClassName**.

Nome	Descrizione
Get Catalog	Get Catalog by class name

```
1 var getCatalog = function() {  
2     CustomAPIService.getCatalogByClassName('RoleType', false, null, null, true).then(  
3         function successCallback(data) {  
4             console.log(data);  
5             viewData.catalogEntries = data;  
6         });  
7 };  
8 registerFunction("getCatalog", getCatalog);
```

2. PREMESSA PER L'UTILIZZO DEI SERVIZI DI INTEGRAZIONE

Di seguito si esplicita la specifica delle chiamate di contesto per la comunicazione tra Form Manager 3 e applicativi esterni (es: Terapie, Silor, Opera). Tale comunicazione ha come presupposto l'autenticazione con token JWT e presuppone l'installazione del modulo **Token Issue Service GPI** per la generazione e la **validazione di un token JWT**.

3. WS PER OTTENERE IL TOKEN DI AUTENTICAZIONE

Il webservice (REST) che permette il recupero del token è raggiungibile all'indirizzo URL [http://\[host\]:\[port\]/TokenIssueService/services/tokenissueservice/login](http://[host]:[port]/TokenIssueService/services/tokenissueservice/login) (ogni installazione ha host e port specifici).

Metodo: POST

Parametri: il metodo prende in input i seguenti parametri

- **system** (String): utente definito nell'installazione del **Token Issue Service** (vedi file **tokenissue.properties** sotto standalone/configuration/gpi del wildfly installato per il servizio) relativo all'applicativo esterno. Lo stesso utente deve risultare censito in NGH con **status FM3**. **Per scopi di test, utilizzare l'utente fm3 con password T3stFM3!**
- **password** (String): password dell'utente suddetto (vedi file **tokenissue.properties** sotto standalone/configuration/gpi del wildfly installato per il servizio)
- **sub** (String): codice utente riferito all'applicativo esterno censito in NGH con **status FM3**. Nel caso di **Terapie** sarà **utentefm.terapie**, nel caso di **Silor** sarà **utentefm.silor**, nel caso di **Opera** sarà **utentefm.opera**.
- **aud** (String): nome fisso sempre pari a **HIS_FORM_MNGMT**
- **values** (oggetto): oggetto del tipo **"values": [{"key": "FM_username", "value": "username"}]** dove username è un valore di testo libero e in genere riferito all'utente censito nell'applicativo chiamante (es: *admin*).
- **assistito** (String): valorizzato sempre come stringa vuota (e ignorato nelle successive chiamate al FM3)

Output: il servizio restituisce un token JWT

Errori gestiti:

- **Autenticazione fallita (system e password)**. Generato in caso di system e/o password errati.
- **Il campo x è obbligatorio**. Generato in caso di assenza di un campo obbligatorio (system, password, sub).
- **Errore durante la fase di ricerca del system. Verificare i dati e riprovare. Di conseguenza non è stato generato token**. Generato in caso di assenza del campo assistito o nel caso di sintassi errata del campo.

Esempio di chiamata

URL: <http://devsrv03.erossi.org:8080/TokenIssueService/services/tokenissueservice/login>

(la chiave per decryptare il token su tscss01 è **GxF=4A?!c\$@17>ul**)

Parametri in input:

```
{
  "system": "fm3",
  "password": "T3stFM3!",
  "sub": "utentefm.terapie",
  "aud": "HIS_FORM_MNGMT",
  "assistito": "",
  "values": [{"key": "FM_username", "value": "admin"}]
}
```

Esempio di risposta

eyJhbGciOiJIbMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4q-HEB1Lr-Ndw.4cS1x8YstZ8e5Fbkn-hdYNNKqYy7udltOqFeukyzBqVOOMNI3hXliGMKSnOKiFsv179jHVIK3JoXOjeAvMsmZAPMY.-vxiAgAimZrJurMVm9MgFuQ

4. WS PER CREARE O AGGIORNARE UN'ISTANZA DI UNA FORM

Il webservice (REST) che permettere di creare o aggiornare un'istanza di una form è **http://[host]:[port]/HIS/rest/form_manager/FMInstance/urlByCodeAndVersionExt** (ogni installazione ha host e una port specifici).

E' possibile istanziare una qualsiasi form configurata in Form Manager 3 (non ci sono vincoli di organizzazione ecc). Una volta creata l'istanza questa viene salvata in Form Manager 3 e il servizio restituisce l'**id dell'istanza della form** da salvare nel db dell'applicazione chiamante per poter recuperare o modificare la form successivamente e l'**url da chiamare** per poter aprire la form stessa con il runtime di Form Manager 3.

Note:

- Se il template creato dal configuratore è in stato DRAFT o INVALID non è possibile creare un'istanza
- Non è possibile aggiornare un'istanza salvata come PUBLISHED
- Non è possibile aggiornare un'istanza con stato DELETED (cancellata con il servizio deleteByIdExt)
- Non è possibile aggiornare un'istanza generata da un altro applicativo chiamante
- Un'istanza creata e visualizzata a runtime ma non salvata rimane in stato NEW.

Metodo: PUT

Parametri dell'header:

- **token** (String): il token ottenuto dal WS di tokenissue suddetto
- **code** (String): codice del template valorizzato nel campo **Codice Form** del configuratore di form
- **version** (Long): versione del template valorizzata nel campo **Versione** del configuratore di form. Se non passo il parametro mi restituirà l'ultima versione della form valida e pubblicata
- **application** (String): nome dell'applicazione chiamante (IE-TERAPIE, IE-OPERA, IE-SILOR...) – **vedi sezione Configurazione applicativi chiamanti del FORM MANAGER 3**
- **formInstanceId** (String **facoltativo**): l'id dell'istanza della form da valorizzare solo in caso di **aggiornamento** . Il servizio restituisce sempre l'**ultima istanza** di form modificata relativa alla versione indicata tra i parametri dell'header. **Se non passo il parametro verrà automaticamente creata una nuova istanza.**
- **forceNewVersion** (Boolean **facoltativo**): true per creare una nuova versione di un'istanza già salvata in Definitivo.

Parametri del body:

- **FMCreateInstanceDTO** così composto:
 - **patient**: oggetto json **obbligatorio per gli ambienti FM3 dove l'integrazione al repository è attiva:**
 1. **patientId** (String) → id salvato su FM3 che potrà essere utilizzato per recuperare i concetti
 2. **firstName** (String)
 3. **lastName** (String)
 4. **sex** (String)
 5. **birthDate** (Date)
 6. **uniqueIdentifier** (String) → identificativo univoco corrispondente all'identificativo salvato sul repository
 7. **phone** (String)
 8. **mobilePhone** (String)
 9. **email** (String)

10. **language** (String)
 11. **deathDate** (Date)
 12. **isDead** (Boolean)
- **initData**: oggetto json da passare alla form come initData e contenenti i valori per preinizializzare la form
 - **runtimeOptions** (oggetto json **facoltativo**): opzioni di **visualizzazione** da settare per la visualizzazione dell'istanza a runtime (saranno valutate solo le opzioni passate):
 1. **hideButtons**: true se si vogliono nascondere tutti i bottoni (eccetto il bottone versioni precedenti)
 2. **hidePreviousVersions**: true se si vuole nascondere il bottone di **Versioni Precedenti**
 3. **hideValidation**: true se si vuole nascondere la parte di **validazione** sotto i bottoni
 4. **hideTitle**: true se si vuole nascondere il **titolo** della form
 5. **hidePublishButton**: true se si vuole nascondere il bottone di **Definitivo**
 6. **hideDraftButton**: true se si vuole nascondere il bottone di **Bozza**
 7. **hidePrintButton**: true se si vuole nascondere il bottone di **Stampa/Anteprima**
 8. **hideExportButton**: true se si vuole nascondere il bottone di **Esporta i dati**
 9. **notifyParent**: true se si vuole che FM3 invii in postMessage messaggi al salvataggio e alla cancellazione dell'istanza (per ulteriori dettagli vedere la sezione PostMessage del documento)
 10. **showNotification**: *NONE* (opzione di default) non mostra nessun messaggio a video; *ERROR* mostra solo i messaggi di errore; *INFO* mostra sia i messaggi di errore che i messaggi di conferma
 11. **readOnly**: true se si vuole aprire l'istanza in modalità di sola lettura. In questo caso i campi di input risulteranno disabilitati e sarà abilitato il solo bottone Annulla.

Output: il servizio restituisce un oggetto json **FMInstanceWrapperDto** così composto:

- **formInstanceValue** (oggetto): istanza della form così composta
 - **id** (Long): identificativo del formInstanceValue associato al formInstancelId
 - **insertUserId** (Long): id dell'utente che ha inserito l'istanza
 - **insertUser** (String): username dell'utente che ha inserito l'istanza
 - **insertTimeUTC** (Date): data di inserimento dell'istanza senza timezone
 - **insertTime** (Date): data di inserimento dell'istanza
 - **updateUserId** (Long): id dell'utente che ha aggiornato l'istanza
 - **updateUser** (String): username dell'utente che ha aggiornato la form
 - **updateTimeUTC** (Date): data di aggiornamento dell'istanza senza timezone
 - **updateTime** (Date): data di aggiornamento dell'istanza
 - **organizationId** (Long): id relativo all'installazione del Form Manager 3
 - **valid** (Boolean): false se l'istanza è stata cancellata logicamente
 - **cancellationTime** (Date): data di cancellazione dell'istanza
 - **cancellationUser** (String): username dell'utente che ha cancellato l'istanza
 - **cancellationUserId** (Long): id dell'utente che ha cancellato l'istanza

- **cancellationTimeUTC** (Date): data di cancellazione dell'istanza senza timezone
- **formInstanceId** (String): id dell'istanza della form che occorre salvare nel DB dell'applicazione chiamante
- **formDocument** (oggetto): oggetto json contenente i dati correnti che valorizzano la form:
 - ◆ **initData** (oggetto): oggetto json passato alla form come initData
 - ◆ **formData** (oggetto): oggetto json contenenti i dati inseriti e salvati nella form
 - ◆ **summaryFieldsData** (array): array dei summary fields definiti (vedere apposita sezione)
 - ◆ **conceptFieldsData** (array): array dei concetti definiti (vedere apposita sezione)
 - ◆ **metadata** (array): array dei metadati definiti per l'invio al repository (vedere apposita sezione)
- **version** (Long): numero di versione dell'istanza della form (non del template). Valorizzato sempre con l'ultima versione. In caso di chiamata con parametro di input forceNewVersion=true su istanza con stato Published, version risulterà incrementata nella risposta.
- **status** (String): può assumere i seguenti valori: NEW, DELETED, DRAFT, PUBLISHED
- **patient** (oggetto): oggetto relativo al paziente
- **urlInstance** (String): url da chiamare per aprire la form con Form Manager 3 (il **runtime**)

Errori gestiti:

- **AccessDeniedException**. Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **Error during find code=x/null and version=1/null**. Generato in caso di chiamata con code null o non passato come parametro, code non esistente, version null o non passato o version non esistente.
- **Parametro application: Input calling application (ssss) not found among available application**. Generato in caso di parametro application non passato o errato.
- **Unrecognized field "xxx" (class it.gpi.his.common.dtos.fm3.FMRuntimeOptionsDto), not marked as ignorable (10 known properties: "hideDraftButton", "readOnly", "notifyParent", "hidePreviousVersions", "hideTitle", "hideExportButton", "hideValidation", "hideButtons", "hidePublishButton", "hidePrintButton"])**. Se passato un campo nelle runtimeOptions non codificato.
- **Cannot create a new FMInstance from an FMTemplate with status: DRAFT**. Se il template è in stato DRAFT, non è possibile creare un'istanza.
- **Cannot create a new FMInstance from an invalid FMTemplate**. Se il template è stato invalidato (perché è stato cancellato o perché è stata creata una nuova versione).
- **Cannot update an instance with current instance value Deleted o Published**. Se l'istanza è in stato PUBLISHED o DELETED, non è possibile aggiornarla da applicativi esterni.
- **Cannot find any FMInstance with ID: xxxx**. Generato quando aggiorno un'istanza con formInstanceId non esistente.
- **An Instance concerning the supplied ID has been found but the application code doesn't match**. Non è possibile aggiornare un'istanza generata da un altro applicativo chiamante.
- **Cannot create a new version of an instance with currentInstanceValue is not Published**. Non è possibile forzare la creazione (tramite parametro forceNewVersion=true) di una nuova versione se l'istanza non è PUBLISHED.
- **Patient is not consistent with repository: could not continue**. Errore generato in caso di form con repository attivo configurato e in caso di servizio senza oggetto patient passato o con oggetto patient mal costruito.
- **Organization parameter FORMMANAGER.FORMMANAGER_REPOSITORY_ENABLED not enabled**. Errore in caso di form con repository attivo configurato ma repository assente o non configurato.

Esempio di chiamata

URL: http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMInstance/urlByCodeAndVersionExt

Parametri in input (HEADER):

- **token:**
*eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZJZqYA.k3t9u3ckIkYr-vCkMTcX3wZhl_NxXc2T_RskLKVoNGYUyMM0Wa-
hibMwnPfkLj6MKpXjKbwidWgZnjPC8gN1gP7d3JnCBGe5Fbkn-
hdYnKqYy7udltOqFeukyzBqVOOMNI3hXliGMKSnOKiFsv179jHVlK3JoXOjeAvMsmZAPMY.vxiAgAi*
- **code:** test_integrazione_05 (vedi immagine sotto)
- **version:** 1 (vedi immagine sotto)
- **application:** IE-TERAPIE
- **formInstanceId:** 5a8ea48288506e31c3ae6202 (parametro da passare solo in caso di aggiornamento)
- **forceNewVersion:** false

Parametri in input (BODY):

```
{
  "patient": {
    "patientId": "1",
    "firstName": "Beatrice",
    "lastName": "Henry",
    "sex": "F",
    "birthDate": "1973-01-09",
    "uniqueIdentifier": "PAT2",
    "phone": "(545)-131-9249",
    "mobilePhone": "(545)-131-9249",
    "email": "beatrice.henry27@example.com",
    "language": "es_ES",
    "deathDate": "",
    "isDead": ""
  },
  "initData": {
    "patient": {
      "name": "Alessandra",
      "surname": "Illuminati"
    }
  },
  "runtimeOptions": {
    "hideButtons": "false",
    "hidePreviousVersions": "false",
    "hideValidation": "false",
    "hideTitle": "false",
    "hidePublishButton": "false",
    "hidePrintButton": "false",
    "hideExportButton": "true",
    "showNotification": "NONE",
    "readOnly": "false"
  }
}
```

I valori passati nell'oggetto json **initData** e utilizzati nella form con l'assegnazione del tipo *formData.field1 = initData.field1* saranno mostrati nella form stessa. L'oggetto **initData** può essere quindi utilizzato per passare alcuni dati alla form come i dati relativi all'utente, alla prescrizione ecc. Per passare dati relativi al paziente, utilizzare l'oggetto **patient**.

Esempio di risposta

```
"formInstanceValue": {
  "id": "5a9f991c88506e54ec98f9a8",
  "insertUserId": 16,
  "insertUser": "FM3",
  "insertTimeUTC": 1520408860668,
  "insertTime": 1520408860668,
  "updateUserId": null,
  "updateUser": null,
  "updateTimeUTC": null,
  "updateTime": null,
  "organizationId": 2,
  "valid": true,
  "cancellationTime": null,
  "cancellationUser": null,
  "cancellationUserId": null,
  "cancellationTimeUTC": null,
  "formInstanceId": "5a9f991c88506e54ec98f9a7",
  "formDocument": {
    "initData": {"patient": {
      "name": "Alessandra",
      "surname": "Altomare"
    }},
    "formData": null,
    "summaryFieldsData": [],
    "conceptFieldsData": [],
    "metaData": []
  },
  "version": 1,
  "status": "NEW",
  "patient": {
    "insertUserId": null,
    "insertUser": null,
    "insertTimeUTC": null,
    "insertTime": null,
    "updateUserId": null,
    "updateUser": null,
    "updateTimeUTC": null,
    "updateTime": null,
    "organizationId": null,
    "valid": null,
    "cancellationTime": null,
    "cancellationUser": null,
    "cancellationUserId": null,
    "cancellationTimeUTC": null,
    "firstName": "Beatrice",
    "lastName": "Henry",
    "sex": "F",
    "birthDate": 95385600000,
    "uniqueIdentifier": "PAT2",
    "phone": "(545)-131-9249",
    "mobilePhone": "(545)-131-9249",
    "email": "beatrice.henry27@example.com",
    "language": "es_ES",
    "deathDate": null,
    "isDead": null,
    "patientId": "1"
  }
},
"urlInstance": "http://devsrv03.erossi.org:8080/HIS/fm3/#/runtime?
application=HIS&=.1Gxmeo_____.&language=it_IT&formInstanceId=5a9f991c88506e54ec98f9a7&fm_username=admin"
}
```

5. WS PER RECUPERARE UNA FORM A PARTIRE DALL'ID DELL'ISTANZA

Il webservice (REST) che permettere di recuperare l'istanza di una form è **http://[host]:[port]/HIS/rest/form_manager/FMInstance/urlByIdExt** (ogni installazione è caratterizzata da un host e una port specifici).

Metodo: PUT

Parametri dell'header:

- **token** (String): il token ottenuto dal WS di tokenissue suddetto
- **formInstanceId** (String): l'id dell'istanza della form salvato nel db dell'applicazione chiamante dopo la creazione della form stessa tramite il ws **urlByCodeAndVersionExt**
- **formInstanceVersion** (String): versione dell'istanza della form che si vuole recuperare

Parametri del body:

- **runtimeOptions** (oggetto json **facoltativo**): oggetto come descritto nel servizio **urlByCodeAndVersionExt**

Output: il servizio restituisce un oggetto analogo all'oggetto restituito dal servizio **urlByCodeAndVersionExt**

Errori gestiti:

- **AccessDeniedException**. Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **Instance id is null, instance having id=null couldn't be fetched**. Generato in caso di parametro **formInstanceId** non presente in chiamata.
- **Error during find FormInstanceDto by instanceId, cannot find FMInstanceDto with instanceId: xxxx**. Generato in caso di **formInstanceId** passato non esistente.
- **FormInstanceValue not found with formInstanceId:xxxx and version:y**. Generato in caso di versione dell'istanza non trovata.

Esempio di chiamata

URL: *http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMInstance/urlByIdExt*

Parametri in input (HEADER):

- **token:**
eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZIZqYA.k3t9u3cklkYr-OjeAvMsmZAPMY.vxiAgAimZrJurMVm9MgFuQ
- **formInstanceId:** 5a8ea48288506e31c3ae6202
- **formInstanceVersion:** 1

Parametri in input (BODY):

```
"runtimeOptions": {  
  "hideButtons": "false",  
  "hidePreviousVersions": "false",  
  "hideValidation": "false",  
  "hideTitle": "false",  
  "hidePublishButton": "false"  
}
```

Esempio di risposta

Risposta analoga a quella descritta per il servizio **urlByCodeAndVersionExt**. In caso di recupero di una versione dell'istanza precedente a quella corrente, l'istanza sarà visibile in modalità read-only.

6. WS PER CANCELLARE UN'ISTANZA DATO L'ID

Il webservice (REST) che permettere di cancellare l'istanza di una form (solo se in stato **DRAFT**) è **http://[host]:[port]/HIS/rest/form_manager/FMInstance/deleteByIdExt** (ogni installazione è caratterizzata da un host e una port specifici).

Metodo: POST

Parametri dell'header:

- **token** (String): il token ottenuto dal WS di tokenissue suddetto
- **formInstanceId** (String): l'id dell'istanza della form salvato nel db dell'applicazione chiamante dopo la creazione della form stessa tramite il ws **urlByCodeAndVersionExt**

Output: il servizio restituisce un oggetto con il campo **formId** corrispondente all'id passato e cancellato e **valid=false**.

Errori gestiti:

- **AccessDeniedException**. Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **Instance id is null, instance having id=null couldn't be fetched**. Generato in caso di parametro formInstanceId non presente in chiamata.
- **Error during find FormInstanceDto by instanceId, cannot find FMInstanceDto with instanceId: xxxx**. Generato in caso di formInstanceId passato non esistente.
- **Cannot delete an Instance with currentInstanceValue Published**. Errore generato in caso di tentativo di cancellazione di un'istanza salvata come definitiva.

Esempio di chiamata

URL: http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMInstance/deleteByIdExt

Parametri in input (HEADER):

- **token:**
*eyJhbGciOiJIbMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZJZqYA.k3t9u3ckIkYr-vCkMTcX3wZhI_NxXc2T_RskLKVoNGYUyMM0Wa-
hibMwnPfkLj6MKpXjKbwidWgZnjPC8gN1gP7d3JnCBGe5Fbkn-
hdYnKqYy7udItOqFeukyzBqVOOMNI3hXliGMKSnOKiFsv179jHVIK3JoXOjeAvMsmZAPMY.vxiAgAimZrJurMVm9MgF
uQ*
- **formInstanceId:** 5a8ea48288506e31c3ae6202

Esempio di risposta

```
{
  "id": "5ab36d3688506e04727495e4",
  "insertUserId": 16,
  "insertUser": "FM3",
  "insertTimeUTC": 1521708342814,
  "insertTime": 1521708342814,
  "updateUserId": 16,
  "updateUser": "FM3",
  "updateTimeUTC": 1521708384990,
  "updateTime": 1521708384990,
  "organizationId": 2,
  "valid": false,
  "cancellationTime": 1521708384989,
  "cancellationUser": "fm3",
  "cancellationUserId": 16,
  "cancellationTimeUTC": 1521704784989,
```

```

"formId": "5a9e4b8688506e59fc203393",
"currentFormInstanceValueId": "5ab36d3688506e04727495e5",
"application": {
  "insertUserId": 2,
  "insertUser": "Gianpaolo Zanella",
  "insertTimeUTC": 1519896132808,
  "insertTime": 1519896132808,
  "updateUserId": null,
  "updateUser": null,
  "updateTimeUTC": null,
  "updateTime": null,
  "organizationId": 2,
  "valid": true,
  "cancellationTime": null,
  "cancellationUser": null,
  "cancellationUserId": null,
  "cancellationTimeUTC": null,
  "application": "IE-TERAPIE"
},
"patient": {
  "insertUserId": null,
  "insertUser": null,
  "insertTimeUTC": null,
  "insertTime": null,
  "updateUserId": null,
  "updateUser": null,
  "updateTimeUTC": null,
  "updateTime": null,
  "organizationId": null,
  "valid": null,
  "cancellationTime": null,
  "cancellationUser": null,
  "cancellationUserId": null,
  "cancellationTimeUTC": null,
  "firstName": "Beatrice",
  "lastName": "Henry",
  "sex": "F",
  "birthDate": 95385600000,
  "uniqueIdentifier": "PAT2",
  "phone": "(545)-131-9249",
  "mobilePhone": "(545)-131-9249",
  "email": "beatrice.henry27@example.com",
  "language": "es_ES",
  "deathDate": null,
  "isDead": null,
  "patientId": "1"
}
}

```

Da osservare che in questo caso sono valorizzati i campi di **cancellazione** e **valid** è **false**:

```

"cancellationTime": 1521708384989,
"cancellationUser": "fm3",
"cancellationUserId": 16,
"cancellationTimeUTC": 1521704784989,

```

compilati anche in caso di chiamata al servizio **urlByIdExt** relativa a istanza con status **DELETED**.

7. WS PER RECUPERARE IL PDF DI STAMPA DELLA FORM

Il webservice (REST) che permettere di recuperare l'istanza di una form è **http://[host]:[port]/HIS/rest/form_manager/FMInstance/printExt** (ogni installazione è caratterizzata da un host e una port specifici).

Metodo: PUT

Parametri dell'header:

- **token** (String): il token ottenuto dal WS di tokenissue suddetto
- **formInstanceId** (String): l'id dell'istanza della form salvato nel db dell'applicazione chiamante dopo la creazione della form stessa tramite il ws **urlByCodeAndVersionExt**
- **getUrl** (boolean): true se tra i risultati si vuole anche l'url da aprire per ottenere il pdf
- **getContent** (boolean): true se tra i risultati si vuole anche il contenuto del pdf
- **application** (String): nome dell'applicazione chiamante (IE-TERAPIE, IE-OPERA, IE-SILOR) – **vedi sezione Configurazione applicativi chiamanti del FORM MANAGER 3**. A seconda dell'application passata verrà lanciata una stampa diversa come da configurazione prevista in Opzioni Template del configuratore della form - **vedi sezione Operazioni sulla form – opzioni template**. In caso di **parametro mancante** sarà lanciato il template di default. In caso di application errata sarà generato errore.

Output: il servizio restituisce un oggetto json DocumentContentDto così composto:

- **xdsEntryUUID** (String): null
- **xsdRepositoryID** (String): null
- **content** (String): contenuto nel pdf
- **mimeType** (String): *application/pdf*
- **externalUrl** (String): url assoluto da utilizzare per aprire il pdf in pagina web. Attenzione: l'url risulta valido solo per 10 secondi (per ottimizzare la cache) a partire dalla sua generazione, quindi dopo questo intervallo di tempo non sarà più possibile accedere al pdf
- **downloadUUID** (String): id del documento associato al report
- **cda2** (String): cda2 generato a partire dal form (**vedi sezione Operazioni sulla form – opzioni template**)

Errori gestiti:

- **AccessDeniedException**. Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **Parameter formInstanceId is missing**. Generato in caso di parametro formInstanceId non passato.
- **No fmInstanceValue found with formInstanceId xxxx**. Generato in caso di formInstanceId passato non esistente.
- **Selected report template null does not exist**. Generato in caso di report non configurato per la form o non esistente.
- **No report generated**. Generato in caso di report non generato correttamente.
- **Parametro application: Input calling application (ssss) not found among available application**. Generato in caso di parametro application errato.

URL: http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMInstance/printExt

Parametri in input (HEADER):

- **token:**
eyJhbGciOiJIbMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZJZqYA.k3t9u3ckIkYr-vCkMTcX3wZhI_NxXc2T_RskLKVoNGYUyMMOWa-hibMwnPfkLj6MKpXjKbwidWgZnjPC8gN1gP7d3JnCBGe5Fbkn-hdYNNkQYy7udltOqFeukyzBqVOOMNI3hXliGMKSnoKiFsv179jHVlK3JoXOjeAvMsmZAPMY.vxiAgAimZrJurMVm9MgFuQ
- **formInstanceId:** 5a8ea48288506e31c3ae6202
- **getUrl:** true
- **getContent:** true
- **application:** IE-TERAPIE

Esempio di risposta

```
{
  "xdsEntryUUID": null,
  "xdsRepositoryID": null,
  "content": "JVBERi0xLjUKJelJz9MKMyAwIG9iago8PC9GaWx0ZXlvRmxhdGVEZWNVZGUVtGVuZ3RoIDExMj4+c3RyZ-
WftCnicK+RyCuEyNIOWMDBTCEnhcg3hCuQyUvACiRoqGAAhILQwMVIlIyESdzNUMDRQCEnj0tAMyQKpRSgxUE-
jORdZkYmGuZ26pgZmymZ2gB02sJ0moAVICUzqWRV5qTazLHQCEdi1nRsUA6BewWAHwVIRYKZW5kc3RyZWftCm-
VuZG9iagoxIDAgb2JqCjw8L1RhYnMvUy9Hcm91cDw8L1MvVHJhbnNwYXJlbnN5L1R5cGUVr3JvdXAvQ1MvRGV2a-
WNlUkdCPj4vQ29udGVudHMgMyAwIFlvVHlwZS9QYWdlL1Jlc291cmNlcw8L0NvbG9yU3BhY2U8PC9DUy9EZXZ-
pY2VSR0I+Pi9Qcm9jU2V0IFsvUERGIC9UZxh0IC9JbWFnZUIGL0ltYWdlQyAvSW1hZ2VJXS9Gb250PDwvRjEgMiAwIFI
+Pj4+L1BhcmVudCA0IDAgUi9NZWRpYUJveFswIDAgNTk1IDg0Ml0+PgplbmRvYmoKNsAwIG9iagpbMSAwIFlvW-
FlaIDAgODUyIDBdCmVuZG9iagoyIDAgb2JqCjw8L1N1YnR5cGUVVHlwZTEvVHlwZS9Gb250L0Jhc2VGB250L0h1b-
HZldGljYS9FbmNvZGluZy9XaW5BbnNpRW5jb2Rpbmc+PgplbmRvYmoKNCAwIG9iagoxPC9LaWRzWzEgMCB-
SXS9UeXBIL1BhZ2VzL0NvdW50IDEvSVRYVCgyLjEuNyK+PgplbmRvYmoKNiAwIG9iagoxPC9OYW1lc1soSlJfUE-
FHRV9BTkNIT1JfMF8xKSA1IDAgU10+PgplbmRvYmoKNyAwIG9iagoxPC9EZXN0cyA2IDAgUj4+CmVuZG9iagoxID-
Agb2JqCjw8L05hbWVzIDcgMCBSL1R5cGUVQ2F0YWxvZy9QYWdlcyA0IDAgUi9WaWV3ZXJQcmVmZXJlbnNlc-
zw8L1ByaW50U2NhbgLuZy9BCHBEZWZhdWx0Pj4+PgplbmRvYmoKOSAwIG9iagoxPC9Nb2REYXRiKEQ6MjAxODAz-
MDExMDA4MjYrMDEnMDAnKS9DcmVhdG9yKEphc3BlcUlcG9ydHMgTGlicmFyeSB2ZXJzaW9uIDYyNS4xKS9Dcm-
VhdGlvbkRhdGUoRDoyMDE4MDMwMTEwMDgxNiswMScwMCcpL1Byb2R1Y2VyKGluZXh0IDluMS43IGJ5IDFUM1-
hUKT4+CmVuZG9iagp4cmVmCjAgMTAKMDAwMDAwMDAwMCA2NTUzNSBmIAowMDAwMDAwMTk0IDAwMDA-
wIG4gCjAwMDAwMDAwMDA0NzAgMDAwMDAgbiAKMDAwMDAwMDAxNSAwMDAwMCBulAowMDAwMDAwNTU4I-
DAwMDAwIG4gCjAwMDAwMDAwMDA0MzUgMDAwMDAgbiAKMDAwMDAwMDYyMSAwMDAwMCBulAowMDAwMDAwM-
DAwNjc1IDAwMDAwIG4gCjAwMDAwMDAwMDA3MDcgMDAwMDAgbiAKMDAwMDAwMDgxMCAwMDAwMC-
BulAp0cmFpbGVyCjw8L0luZm8qOSAwIFlvSUQgWzw1NjEyYzAxN2YwZjdlNzMyY2U2ZGnmYzczMTM5Mzk1YT48-
MjE2OTNmNWUyNmMxOTRlNzYyMjY3NDcwZjRjN2IxZjY+XS9Sb290IDggMCBSL1NpemUgMTA+PgpdZGFydHh-
ZWYKOTc3CiU1RU9GCg==",
  "mimeType": "application/pdf",
  "externalUrl": "http://devsrv03.erossi.org:8080/HIS/rest/repository/document/contentDownload?
download=false&uuid=adbb29f3-0aee-41d4-972a-0be6a192e29b&fileName=adbb29f3-0aee-41d4-972a-
0be6a192e29b.pdf&mimeType=application/pdf",
  "downloadUUID": "a0f9c107-0b4f-4d75-a70d-0f017e3761da",
  "cda2": null
}
```

8. WS PER PUBBLICARE UN'ISTANZA DATO L'ID

Il webservice (REST) che permettere di cancellare l'istanza di una form in stato NEW o DRAFT è **http://[host]:[port]/HIS/rest/form_manager/FMInstance/publishByIdExt** (ogni installazione è caratterizzata da un host e una port specifici).

Il servizio permette di pubblicare un'istanza (cambio stato in DEFINITIVE) solo se l'istanza è in stato NEW o DRAFT.

Metodo: POST

Parametri dell'header:

- **token** (String): il token ottenuto dal WS di tokenissue suddetto
- **formInstanceId** (String): l'id dell'istanza della form salvato nel db dell'applicazione chiamante dopo la creazione della form stessa tramite il ws **urlByCodeAndVersionExt**

Errori gestiti:

- **AccessDeniedException.** Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **Instance id is null, instance having id=null couldn't be fetched.** Generato in caso di parametro formInstanceId non presente in chiamata.
- **Error during find FormInstanceDto by instanceId, cannot find FMInstanceDto with instanceId: xxxx.** Generato in caso di formInstanceId passato non esistente.
- **Cannot publish an Instance with currentInstanceValue PUBLISHED.** Generato in caso di servizio chiamato su istanza già pubblicata.
- **Cannot publish an Instance with currentInstanceValue DELETED.** Generato in caso di servizio chiamato su istanza cancellata

Esempio di chiamata

URL: http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMInstance/publishByIdExt

Parametri in input (HEADER):

- **token:**
*eyJhbGciOiJIbMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZJZqYA.k3t9u3ckIkYr-vCkMTcX3wZhI_NxXc2T_RskLKVoNGYUyMM0Wa-
hibMwnPfkLj6MKpXjKbwidWgZnjPC8gN1gP7d3JnCBGe5Fbkn-
hdYnKqYy7udltOqFeukyzBqVOOMNI3hXliGMSnOkIfsv179jHVIK3JoXOjeAvMsmZAPMY.vxiAgAimZrJurMVm9MgF
uQ*
- **formInstanceId:** 5a8ea48288506e31c3ae6202

Esempio di risposta

```
{  
  "id": "5ab371a188506e04727495eb",  
  "insertUserId": 16,  
  "insertUser": "FM3",  
  "insertTimeUTC": 1521709473329,  
  "insertTime": 1521709473329,  
  "updateUserId": 16,  
  "updateUser": "FM3",  
  "updateTimeUTC": 1521709606688,  
  "updateTime": 1521709606688,  
  "organizationId": 2,  
  "valid": true,  
  "cancellationTime": null,  
}
```

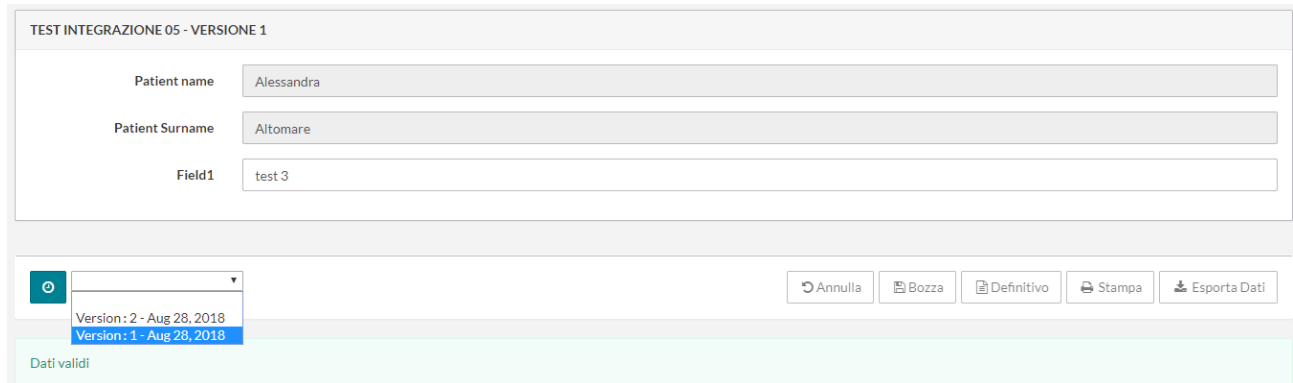
```

"cancellationUser": null,
"cancellationUserId": null,
"cancellationTimeUTC": null,
"formId": "5a9e4b8688506e59fc203393",
"currentFormInstanceValueId": "5ab371a188506e04727495ec",
"application": {
  "insertUserId": 2,
  "insertUser": "Gianpaolo Zanella",
  "insertTimeUTC": 1519896132808,
  "insertTime": 1519896132808,
  "updateUserId": null,
  "updateUser": null,
  "updateTimeUTC": null,
  "updateTime": null,
  "organizationId": 2,
  "valid": true,
  "cancellationTime": null,
  "cancellationUser": null,
  "cancellationUserId": null,
  "cancellationTimeUTC": null,
  "application": "IE-TERAPIE"
},
"patient": {
  "insertUserId": null,
  "insertUser": null,
  "insertTimeUTC": null,
  "insertTime": null,
  "updateUserId": null,
  "updateUser": null,
  "updateTimeUTC": null,
  "updateTime": null,
  "organizationId": null,
  "valid": null,
  "cancellationTime": null,
  "cancellationUser": null,
  "cancellationUserId": null,
  "cancellationTimeUTC": null,
  "firstName": "Beatrice",
  "lastName": "Henry",
  "sex": "F",
  "birthDate": 95385600000,
  "uniqueIdentifier": "PAT2",
  "phone": "(545)-131-9249",
  "mobilePhone": "(545)-131-9249",
  "email": "beatrice.henry27@example.com",
  "language": "es_ES",
  "deathDate": null,
  "isDead": null,
  "patientId": "1"
}
}

```


9. RUNTIME

Utilizzando l'url passato in **urlInstance** dai precedenti servizi è possibile aprire l'istanza della form a **runtime**. E' possibile aprire l'istanza anche in un **frame**. Il runtime si presenta come nell'immagine sopra



Attenzione: in caso di errore 401 nell'apertura del runtime, o in caso di visualizzazione vuota, cancellare il local storage del browser o utilizzare la navigazione in incognito.

I valori passati nell'oggetto initData e utilizzati nella form sono visibili (es: vedi patient name e surname).

```
"initData": {  
  "patient": {  
    "name": "Alessandra",  
    "surname": "Altomare"  
  }  
}
```

Per **salvare** i dati inseriti nella form è possibile cliccare su:

- **Bozza:** salvataggio istanza in bozza (DRAFT); possibile cancellare/modificare l'istanza nelle chiamate successive. Se l'istanza è salvata come Bozza, il bottone Annulla risulterà disabilitato.
- **Definitivo:** salvataggio definitivo dell'istanza (PUBLISHED); non sarà più possibile cancellare/modificare l'istanza. Se l'istanza è salvata come PUBLISHED saranno disabilitati i bottoni di Annulla, Bozza e Definitivo.

I **bottoni** sono **visibili** se la form è **valida**, cioè se sono stati compilati correttamente tutti i campi obbligatori e/o se sono soddisfatti tutti i controlli di validazione. In caso contrario comparirà la seguente dicitura in basso e solo il bottone Annulla sarà visibile

Dati non validi

Se l'istanza è stata **cancellata** con il servizio deleteByIdExt (stato=DELETED) i campi saranno disabilitati e i bottoni di salvataggio e di annulla saranno disabilitati.

Per annullare un'istanza appena creata (stato NEW) è possibile cliccare su **Annulla**. Cliccando comparirà il seguente messaggio di notifica:

ANNULLAMENTO MODIFICHE ALLA SCHEDA

Questa operazione scarcerà le informazioni correnti e cancellerà i dati dal database. Vuoi continuare?

OK Annulla

Confermando l'annullamento, comparirà sulla pagina un messaggio di conferma e tutti gli altri bottoni saranno non visibili.

La form è stata eliminata con successo , si consiglia di chiudere la finestra

Cliccando sul bottone iconico dell'orologio (**Mostra versioni precedenti**) si potranno visualizzare i dati dell'istanza relativi a versioni precedenti della stessa. I dati sulla versione sono esplicitati nel menu a tendina (numero versione – data di salvataggio). Ricliccando sul bottone iconico, si ritorna all'ultima versione (quella correntemente istanziata).

Version : 2 - Aug 28, 2018
Version : 1 - Aug 28, 2018

Il bottone **Esporta Dati** (se configurato come visibile) consente di esportare i dati della form in un json (funzione utile per controllare cosa viene proposto in stampa).

Se alla form è stato aggiunto un report in configurazione, sarà possibile stampare l'istanza della form cliccando sul bottone **Stampa**.

Esempio di stampa

Configurata la seguente form dove per Field1 ho ng-model=formData.field1:

TEST FORM CON REPORT

Field1

Enter a field value

The value is:

e associando il seguente report di stampa:

```

<?xml version="1.0" encoding="UTF-8"?>
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports http://jasperreports.sourceforge.net/xsd/jasperreport.xsd"
="test_fm3" language="groovy" pageWidth="595" pageHeight="842" columnWidth="555" leftMargin="20" rightMargin="20" topMargin="20"
bottomMargin="20" uuid="9b751a4a-d6d6-4e0a-ac83-3d3eld8e27df">
  <property name="ireport.zoom" value="1.0"/>
  <property name="ireport.x" value="0"/>
  <property name="ireport.y" value="0"/>
  <style name="Heading 1" fontSize="16" isBold="true"/>
  <field name="field1" class="java.lang.String">
    <fieldDescription><![CDATA[formDocument.formData.field1]]></fieldDescription>
  </field>
  <background>
    <band splitType="Stretch"/>
  </background>
  <title>
    <band height="79" splitType="Stretch"/>
  </title>
  <pageHeader>
    <band height="35" splitType="Stretch"/>
  </pageHeader>
  <columnHeader>
    <band height="61" splitType="Stretch"/>
  </columnHeader>
  <detail>
    <band height="16">
      <textField>
        <reportElement x="414" y="0" width="121" height="15"/>
        <textElement textAlignment="Center" verticalAlignment="Middle">
          <font size="9"/>
        </textElement>
        <textFieldExpression class="java.lang.String">
          <![CDATA[${F{field1}}]>
        </textFieldExpression>
      </textField>
    </band>
  </detail>
  <columnFooter>
    <band height="45" splitType="Stretch"/>
  </columnFooter>
  <pageFooter>
    <band height="54" splitType="Stretch"/>
  </pageFooter>
  <summary>
    <band height="42" splitType="Stretch"/>
  </summary>
</jasperReport>

```

se nel report referenzio il campo con la sintassi utilizzata in configurazione:

<fieldDescription><![CDATA[formDocument.formData.field1]]></fieldDescription>

in stampa otterrò il valore inserito nella form per il campo field1

	Test per la stampa
--	--------------------

Per visualizzare a runtime i **messaggi di errore** (solo da NGH) settare il parametro di configurazione **Check Changes on savings form** in Admin – Organization Parameters – User Management.

10. POSTMESSAGE

I `PostMessage`, standard **javascript**, sono utilizzati per far comunicare l'applicazione chiamante e FM3 (che solitamente è dentro un `iframe` dell'applicazione chiamante) tramite **messaggi**. Possono quindi essere utilizzati dall'applicativo chiamante (purché censito tra gli applicativi come descritto nel paragrafo **Configurazione applicativi chiamanti**) per **eseguire** determinate azioni sull'istanza aperta con il FM3 o per **ricevere messaggi** in funzione delle azioni eseguite su FM3.

Documentazione: <https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage>

Caso 1: `PostMessage` inviato da applicativo chiamante a FM3

Caso in cui l'applicativo chiamante si integra con FM3 senza sfruttare le funzionalità e i bottoni presenti nel runtime di FM3 ma governando il salvataggio e l'annulla direttamente dall'applicativo chiamante. **In questo caso la `postMessage` è inviata dall'applicativo chiamante a FM3.**

Scenario di esempio

L'applicativo chiamante invia al FM3 un messaggio per il **salvataggio in bozza** (analogo al click sul bottone Bozza del runtime) e quando FM3 lo riceve, salva e risponde al mittente con un messaggio del tipo 'OK'. L'applicazione chiamante a questo punto sa che la form è stata salvata in bozza e può continuare con la propria logica applicativa.

Esempio di codice sull'applicativo chiamante

```
$scope.saveForm = function(type) {  
    var data = {};  
    data.esito = "OK"  
    data.buttonType = type;  
    data.application = "IE-TERAPIE"  
    var target_url = new URL($scope.instance.url).origin;  
    var frame = document.getElementById('formManagerContentPage');  
    FormService.postMessage(data, target_url, frame.contentWindow, $window);  
}
```

Parametri da settare

- **esito:** Messaggio di risposta fornita dopo la chiamata in `postMessage` effettuata dall'applicativo chiamante
- **buttonType:**
 - **'definitive'** - Salva in Published
 - **'draft'** - Salva in Draft
- **application:** nome dell'applicativo chiamante (come configurato su FM3). In caso di applicazione non censita in FM3 ritorna l'errore *The calling application is not correct.*
- **targetUrl:** URL dell'istanza da aprire con FM3 (restituita dai servizi descritti nei paragrafi precedenti)
- **frame.getContentWindow:** `iFrame` dove viene aperta la URL dell'istanza di FM3

```

var _postMessage = function(message, target_url, target, window) {
  if (!target_url) {
    return;
  }
  target = target || parent; // default to parent
  if (window['postMessage']) {
    // the browser supports window.postMessage, so call it with a targetOrigin
    // set appropriately, based on the target_url parameter.
    target['postMessage'](message, target_url.replace(/([^\:]|\V|[/^\V]+).*/ , '$1'));
  }
  else if (target_url) {
    // the browser does not support window.postMessage, so use the window.location.hash fragment hack
    target.location = target_url.replace(/#.$/, '') +
      '#' + (+new Date) + (cache_bust++) + '&' +
      message;
  }
};

```

NOTA: In caso di dati non validi, il salvataggio sarà interrotto.

Caso 2: PostMessage inviato da FM3 ad applicativo chiamante

Caso in cui l'applicativo chiamante si integra con FM3 sfruttando i bottoni del runtime di FM3 per il salvataggio e l'annulla. **In questo caso la postMessage è inviata da FM3 all'applicativo chiamante.** Per settare l'invio di postMessage da FM3 all'applicativo chiamante, nella chiamata ai servizi rest che restituiscono l'url da aprire a runtime, occorre settare tra le **runtimeOptions**, **notifyParent = true**.

Scenario di esempio

L'utente apre l'istanza di FM3 e clicca sui bottoni di Salva (bozza o definitivo) o sul bottone Annulla. In funzione di una di queste azioni, FM3 invia un messaggio all'applicazione chiamante che riporta informazioni sull'azione eseguita, l'esito dell'azione e l'id dell'istanza. Ricevuto questo messaggio l'applicativo chiamante sa che l'azione è stata eseguita sull'istanza e può continuare con la propria logica applicativa.

Esempio di postMessage inviato da FM3 al salvataggio dell'istanza da intercettare sull'applicativo chiamante

Se l'utente clicca sul bottone Definitivo all'apertura dell'istanza di FM3, FM3 in seguito al click esegue il salvataggio e risponde all'applicativo chiamante con il seguente messaggio:

```

{
  data.esito = "OK",
  data.error = "",
  data.application = "FM3",
  data.formInstanceId = "5ab371a188506e04727495ec",
  data.buttonType = "definitive"
}

```

Parametri inviati da FM3 in postMessage dopo le azioni di salvataggio o annulla

- **esito:** "OK" se il salvataggio o la cancellazione dell'istanza è andata a buon fine, "KO" altrimenti
- **buttonType:**
 - **'definitive'** – se l'utente ha cliccato su Definitivo
 - **'draft'** - se l'utente ha cliccato su Bozza
 - **'cancel'** - se l'utente ha cliccato su Annulla
 - **'print'** – se l'utente clicca su Stampa
- **application:** nome dell'applicativo chiamante, in questo caso sempre FM3
- **formInstanceId:** id dell'istanza aperta in FM3 a cui è riferita l'azione, null in caso di esito KO
- **error:** errore (Stringa) ritornato in caso di esito KO

Esempio di Json inviato da FM3 in caso di esito OK:

```
data: {  
  application: "FM3",  
  buttonType: "print",  
  error: null,  
  esito: "OK",  
  formInstanceId: 5ab371a188506e04727495ec  
}
```

Esempio di Json inviato da FM3 in caso di esito KO:

(nell'esempio specifico l'errore è dovuto alla mancata configurazione del template sulla form)

```
data: {  
  application: "FM3",  
  buttonType: "print",  
  error: "Error....",  
  esito: "KO",  
  formInstanceId: null  
}
```

11. CONCETTI

Nella definizione di una form nel configuratore è possibile definire alcuni campi come **concept**. I concetti sono da intendersi come informazioni importanti relative alla form, che ad esempio consentiranno di interrogare il FormManager ricercando le form per concetti.

Configuratore dei concetti

Per definire i concetti, esiste una sezione apposita del configuratore chiamata **CONCETTI**.



Accedendo a questa sezione, si visualizza la **lista** dei concetti definiti:

Visualizzati 3/7 Elementi

Codice concetto	
peso	✕
paziente.peso	
patient.weigth	

Mostra

E' possibile

- **ricercare** un concetto
- **ordinare** i concetti per codice
- **eliminare** un concetto (sfruttando il bottone apposito della seconda colonna). L'eliminazione invalida il concetto, non permettendone più l'utilizzo
- visualizzare i concetti cancellati in lista cliccando sul bottone **Mostra cancellati**

Per definire un **nuovo concetto** occorre cliccare sul bottone in alto a destra della sezione:



La maschera di **creazione** del concetto richiede di definire

- **Codice concetto**: codice con il quale il concetto sarà referenziato nella form
- **Lingua**: per ogni lingua (definite nella lista del campo Lingua) è possibile aggiungere un valore da associare come label al nuovo concetto definito

Per salvare il nuovo concetto cliccare su **Aggiungi**.

Concetti

Codice concetto

Lingua

Value

+

Lingua

Value

Aggiungi

Annulla

Analogamente per **modificare** un concetto, occorre cliccare sul record corrispondente della lista e cliccare sul bottone **Salva** della maschera di modifica:

Concetti

Codice concetto

Lingua

Lingua	Valore	
it_IT	peso	x

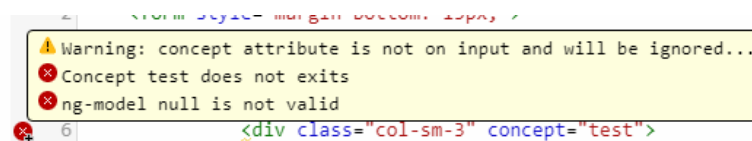
E' possibile definire anche un'**alberatura logica** dei concetti definendo ad esempio il seguente concetto: **patient.weight** dove patient è il concetto padre e weight è il concetto figlio.

Utilizzo dei concetti nella form

Dopo aver definito i concetti è possibile utilizzarli nell'editor HTML, associando l'attributo **concept** al campo di tipo **input** che si vuole referenziare. Il **valore dell'attributo** è il **codice del concetto**, definito come nella sezione concetti (es: peso).

```
<div class="col-sm-3">
  <label>Peso</label>
  <input type="text" concept="peso" summary ng-model="formData.patient.weight">
</div>
<div class="col-sm-3">
  <label>Altezza</label>
  <input type="text" concept="altezza" ng-model="formData.patient.height">
</div>
```

Se si utilizza un codice concetto non definito, l'Editor HTML segnalerà l'errore **Concept <name> does not exists** e il sistema non consentirà il salvataggio della form. Se l'attributo concept è assegnato a un campo diverso da input, il sistema indicherà un **warning** e l'assegnazione del concetto sarà ignorata. Se al campo di tipo input non è indicata la direttiva ng-model con il rispettivo valore, il sistema segnalerà l'errore: **ng-model null is not valid** (l'assegnazione del concetto richiede che l'ng-model sia valorizzato).



A destra della pagina è visibile un **riepilogo dei concetti** definiti per la form. Per ogni concetto sono riportati i bundles definiti nella varie lingue nel configuratore.

Mostra JSON dei concetti	
Mostra JSON dei concetti	
altezza	
it_IT	altezza
peso	
it_IT	peso

Definizione dei concetti tramite funzione Javascript

Alternativamente al configuratore dei concetti è possibile salvare un concetto utilizzando la funzione JavaScript `registerConcept` presente nella libreria Javascript: **`registerConcept("conceptName","conceptValue")`**, dove il `conceptName` è il codice del concetto che si vuole salvare e il `conceptValue` è il rispettivo valore.

Valorizzazione dei concetti nei servizi chiamati da applicativi esterni

Nel recupero del json associato alla form tramite i servizi `urlByIdExt` e `urlByCodeAndVersionExt`, i concetti saranno associati al `formDocument` con un oggetto json chiamato **`conceptFieldsData`** (vedi esempio sotto). L'oggetto si comporrà dei vari concept definiti, per ogni concetto è riportato il **codice**, il **valore** e le **label** associate nelle diverse lingue.

```
"formDocument": {
  "initData": { ... },
  "formData": { ... },
  "conceptFieldsData": [
    {
      "key": "peso",
      "value": "32",
      "bundles": {"it_IT": "peso"}
    },
    {
      "key": "peso",
      "value": "45",
      "bundles": {"it_IT": "peso"}
    },
    {
      "key": "temperature",
      "value": "46",
      "bundles": {
        "it_IT": "Temperatura corporea",
        "en_GB": "Body temperature"
      }
    }
  ]
}
```

Recupero dei concetti nella form

Sfruttando l'editor JavaScript è possibile chiamare i servizi REST sui concetti all'interno della form stessa, per la valorizzazione di determinati campi. Nella libreria JavaScript i componenti presenti sono

- `FIND CONCEPT VALUE` (analogo del **`findFMConceptValueExt`** spiegato in seguito)
- `FIND CONCEPT INSTANCEVALUE` (analogo del **`findFMConceptInstanceValueExt`** spiegato in seguito)
- `FIND CONCEPTS BY PATIENTS` (analogo del **`findFMConceptValueByPatientIdExt`** spiegato in seguito)
- `GET LAST CONCEPT VALUE` (analogo del **`getLastFMConceptValueExt`** spiegato in seguito)

Il loro funzionamento è analogo a quello descritto nei prossimi paragrafi.

12. WS PER RECUPERARE VALORI DEI CONCETTI E ID DELLE ISTANZE TRAMITE CONCETTO

Il webservice (REST) che permettere di recuperare i valori dei concetti e le rispettive istanze delle form associate tramite codice del concetto è `http://[host]:[port]/HIS/rest/form_manager/FMConcept/findFMConceptValueExt` (ogni installazione è caratterizzata da un host e una port specifici).

NOTA: il servizio restituisce solo i valori afferenti alle **istanze** delle form in stato **PUBLISHED** a meno di non settare in chiamata il parametro `includeDraft` a true.

Metodo: PUT

Parametri dell'header:

- **token** (String): **OBBLIGATORIO**. il token ottenuto dal WS di tokenissue suddetto.
- **concept** (String): **OBBLIGATORIO**. Codice del concetto che si vuole ricercare. E' possibile ricercare anche per **concetto padre** con una sintassi del tipo `patient.*` dove `patient` è il concetto padre.
- **templateName** (String): nome del template della form dove si vuole ricercare il concetto. Se cerco solo per `templateName` recupererò tutte le istanze di quel template, riferite a tutti i pazienti e a tutte le date.
- **version** (Long): versione del template della form dove si vuole ricercare il concetto. Da specificare solo nel caso in cui sia specificato anche il parametro `templateName`. Nel caso in cui sia specificato il `templateName` senza `version`, sarà restituita l'**ultima versione** del template.
- **from** (String): data dalla quale si vogliono recuperare i concetti (da esprimere nel formato YYYY-MM-DD)
- **to** (String): data fino alla quale si vogliono recuperare i concetti (da esprimere nel formato YYYY-MM-DD)
- **patientId** (String): id del paziente sul quale si vogliono recuperare i concetti. Se cerco solo per `patientId` recupererò tutte le istanze compilate su quel paziente, riferite a tutti i template e a tutte le date.
- **includeDraft** (Boolean): se true, nei risultati include anche i valori delle istanze salvate in bozza

Parametri del body:

- **templateList** (array): array dei nomi dei template dove si vuole ricercare il concetto. Alternativo alla specifica di `templateName` e `version`

NOTE:

E' obbligatorio specificare **almeno un parametro** tra `templateName` (o `templateList`), `from`, `to`, `patientId` per poter effettuare la ricerca.

E' possibile specificare uno solo dei parametri **from** o **to** per filtrare sulle date (**filtro inclusivo** sul giorno). Se cerco solo per data recupererò tutte le istanze compilate nel range di date, riferite a tutti i template e a tutti i pazienti.

Output: il servizio restituisce un **array di oggetti json** dove ogni oggetto risulta così composto:

- **concept:** codice del concetto ricercato
- **value:** valore del concetto nell'istanza della form corrispondente
- **date:** timestamp della data in cui l'istanza è stata salvataggio
- **instanceValueId:** id dell'istanza della form a cui il concetto fa riferimento

- **code:** codice del template a cui fa riferimento l'istanza.
- **version:** versione del template a cui fa riferimento l'istanza. Valorizzato solo nel caso in cui il servizio sia chiamato con il parametro di header `templateName`
- **status:** stato dell'istanza (draft, published)
- **instanceValues:** per questo servizio sempre null

Errori gestiti:

- **AccessDeniedException.** Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **javax.validation.ConstraintViolationException: 1 constraint violation(s) occurred during method validation.** Constructor or Method: public java.util.List. Generato in caso di token o concept mancante (parametri obbligatori).
- **Error during find FMConceptValue , cannot find any template with code concept.** Generato in caso di presenza, nell'array `templateList`, di codici di template non esistenti (è sufficiente che uno solo dei codici sia errato perché l'errore sia generato)
- **Error during find FMConceptValue , need to specify at least 1 parameter between range of date, template or patient id.** Generato in caso di mancata definizione di almeno un parametro tra `templateName`, `from`, `to`, `patientId`.
- **Error during find FMConceptValue, cannot find any template with code xxx and version yy.** Generato in caso di parametro specificato in maniera errata (ad esempio versione o template non esistenti).
- **Error during find FMConceptValue, cannot specify a version without a template name.** Generato in caso di parametro `version` specificato senza parametro `template`.
- **Error during find FMConceptValue ,cannot specify a version for a templateList.** Generato in caso di parametro `version` specificato quando è settato il parametro `templateList`.
- **Error during find FMConceptValue, fromDate is note before toDate.** Generato in caso di data specificata nel `from` successiva alla data specificata in `to`.

Esempio di chiamata

URL: http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMConcept/findFMConceptValueExt

Parametri in input (HEADER):

- **token:** `eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHI`
- **concept:** peso
- **templateName:** test_concept_summary_2
- **version:** 1
- **from:** 2018-05-10
- **to:** 2018-05-11
- **patientId:** 444
- **includeDraft:** false
- **status:** stato dell'istanza (draft, published)

Parametri del body:

- `["conley-concept-summary", "conceptServiceTest"]`

Esempio di risposta

```
{ "conceptValues": [  
  {  
    "concept": "peso",  
    "value": "4",  
    "date": 1526028961976,  
    "instanceValueId": "5af55aa188506e40cb0cf800",  
    "code": "test_concept_summary_2",  
    "version": 2,  
    "status": "draft"  
  },  
  {  
    "concept": "peso",  
    "value": "5",  
    "date": 1526028961976,  
    "instanceValueId": "5af55aa188506e40cb0cf800",  
    "code": "test_concept_summary_2",  
    "version": 2,  
    "status": "draft"  
  },  
  {  
    "concept": "peso",  
    "value": "67",  
    "date": 1526030205633,  
    "instanceValueId": "5af55f7d88506e40cb0cf804",  
    "code": "test_concept_summary_2",  
    "version": ,  
    "status": "published"  
  },  
  {  
    "concept": "peso",  
    "value": "68",  
    "date": 1526030205633,  
    "instanceValueId": "5af55f7d88506e40cb0cf804",  
    "code": "test_concept_summary_2",  
    "version": 2,  
    "status": "published"  
  }  
],  
"instanceValues": nullable  
}
```

Se non ci sono risultati viene restituito il seguente oggetto json

```
{  
  "conceptValues": [],  
  "instanceValues": null,  
}
```

Chiamata del servizio all'interno della form

Questo servizio può essere richiamato anche all'interno della form, utilizzando nell'editor JavaScript la funzione presente in libreria e chiamata **FIND CONCEPT VALUE**:

```

var findFMConceptValue = function() {
    CustomAPIService.findFMConceptValue("conceptName","templateName","templateVersion","Date : from","Date :
to","patientId","includeDraft").then(
    function successCallback(data) {
        console.log(data);
    },
    function errorCallback(error) {
        console.log(error);
    });
};

```

Questa funzione consente di ottenere lo stesso risultato sopra illustrato e utilizza gli stessi parametri di input a eccezione del token.

13. WS PER RECUPERARE LE ISTANZE TRAMITE CONCETTO

Il webservice (REST) che permettere di recuperare le istanze delle form associate al concetto ricercato è **http://[host]:[port]/HIS/rest/form_manager/FMConcept/findFMConceptInstanceValueExt** (ogni installazione è caratterizzata da un host e una port specifici).

NOTA: il servizio restituisce solo i valori afferenti alle **istanze** delle form in stato **PUBLISHED** a meno di non settare in chiamata il parametro includeDraft a true.

Metodo: PUT

Parametri dell'header:

- **token** (String): **OBBLIGATORIO**. il token ottenuto dal WS di tokenissue suddetto.
- **concept** (String): **OBBLIGATORIO**. Codice del concetto che si vuole ricercare. E' possibile ricercare anche per **concetto padre** con una sintassi del tipo patient.* dove patient è il concetto padre.
- **templateName** (String): nome del template della form dove si vuole ricercare il concetto. Se cerco solo per templateName recupererò tutte le istanze di quel template, riferite a tutti i pazienti e a tutte le date.
- **version** (Long): versione del template della form dove si vuole ricercare il concetto. Da specificare solo nel caso in cui sia specificato anche il parametro templateName. Nel caso in cui sia specificato il templateName senza version, sarà restituita l'**ultima versione** del template.
- **from** (String): data dalla quale si vogliono recuperare i concetti (da esprimere nel formato YYYY-MM-DD)
- **to** (String): data fino alla quale si vogliono recuperare i concetti (da esprimere nel formato YYYY-MM-DD)
- **patientId** (String): id del paziente sul quale si vogliono recuperare i concetti. Se cerco solo per patientId recupererò tutte le istanze compilate su quel paziente, riferite a tutti i template e a tutte le date.
- **includeDraft** (Boolean): se true, nei risultati include anche i valori delle istanze salvate in bozza

Parametri del body:

- **templateList** (array): array dei nomi dei template dove si vuole ricercare il concetto. Alternativo alla specifica di templateName e version

NOTE:

E' obbligatorio specificare **almeno un parametro** tra templateName (o templateList), from, to, patientId per poter effettuare la ricerca.

E' possibile specificare uno solo dei parametri **from** o **to** per filtrare sulle date (**filtro inclusivo** sul giorno). Se cerco solo per data recupererò tutte le istanze compilate nel range di date, riferite a tutti i template e a tutti i pazienti.

Output: il servizio restituisce un **array delle istanze delle form (oggetti json)** associate al concetto ricercato

Errori gestiti:

- **AccessDeniedException.** Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **javax.validation.ConstraintViolationException: 1 constraint violation(s) occurred during method validation.** Constructor or Method: **public java.util.List.** Generato in caso token o concept mancante (parametri obbligatori).
- **Error during find FMConceptValue , need to specify at least 1 parameter between range of date, template or patient id.** Generato in caso di mancata definizione di almeno un parametro tra `templateName`, `from`, `to`, `patientId`.
- **Error during find FMConceptValue, cannot find any template with code xxx and version yy.** Generato in caso parametro specificato in maniera errata (ad esempio versione o template non esistenti).
- **Error during find FMConceptValue, cannot specify a version without a template name.** Generato in caso di parametro version specificato senza parametro template.
- **Error during find FMConceptValue, fromDate is note before toDate.** Generato in caso di data specificata nel `from` successiva alla data specificata in `to`.
- **Error during find FMConceptValue ,cannot specify a version for a templateList.** Generato in caso di parametro version specificato quando è settato il parametro `templateList`.
- **Error during find FMConceptValue , cannot find any template with code concept.** Generato in caso di presenza, nell'array `templateList`, di codici di template non esistenti (è sufficiente che uno solo dei codici sia errato perché l'errore sia generato)

Esempio di chiamata

URL: http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMConcept/findFMConceptInstanceValueExt

Parametri in input (HEADER):

- **token:**
eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZIZqYA.k3t9u3ckIkYr-vCKMTcX3wZh_NxXc2T_RskLKVoNGY
- **concept:** peso
- **templateName:** test_concept_summary_2
- **version:** 1
- **from:** 2018-05-10
- **to:** 2018-05-11
- **patientId:** 444
- **includeDraft:** false

Parametri del body:

- ["conley-concept-summary", "conceptServiceTest"]

Esempio di risposta

```
"conceptValues": null,
"instanceValues": [ {
  "id": "5af55aa188506e40cb0cf800",
  "insertUserId": 16,
  "insertUser": "FM3",
  "insertTimeUTC": 1526028961976,
  "insertTime": 1526028961976,
  "updateUserId": 16,
  "updateUser": "FM3",
  "updateTimeUTC": 1526029002347,
  "updateTime": 1526029002347,
  "organizationId": 2,
  "valid": true,
  "cancellationTime": null,
  "cancellationUser": null,
  "cancellationUserId": null,
  "cancellationTimeUTC": null,
  "formInstanceId": "5af55aa188506e40cb0cf7ff",
  "formDocument": {
    "initData": {"patient": {
      "name": "Alessandra",
      "surname": "Altomare"
    }},
    "formData": {"patient": {
      "weigh": "4",
      "weigh2": "5",
      "heigh": "6",
      "temperature": "7",
      "pain": "no",
      "pa": "10"
    }},
    "summaryFieldsData": [
      {
        "key": "peso",
        "value": "4",
        "isConcept": true,
        "bundles": {"it_IT": "peso"}
      },
      {
        "key": "pain",
        "value": "no",
        "isConcept": false,
        "bundles": {"it_IT": "dolore"}
      }
    ],
    "conceptFieldsData": [
      {
        "key": "peso",
        "value": "4",
        "bundles": {"it_IT": "peso"}
      },
      {
        "key": "peso",
        "value": "5",
        "bundles": {"it_IT": "peso"}
      }
    ]
  }
}
```

```

        {
          "key": "altezza",
          "value": "6",
          "bundles": {"it_IT": "altezza"}
        },
        {
          "key": "temperature",
          "value": "7",
          "bundles": {
            "it_IT": "Temperatura corporea",
            "en_GB": "Body temperature"
          }
        },
        {
          "key": "patient.pa",
          "value": "10",
          "bundles": {"it_IT": "pressione arteriosa"}
        }
      ]
    },
    "version": 1,
    "status": "DRAFT",
    "patient": {
      "insertUserId": null,
      "insertUser": null,
      "insertTimeUTC": null,
      "insertTime": null,
      "updateUserId": null,
      "updateUser": null,
      "updateTimeUTC": null,
      "updateTime": null,
      "organizationId": null,
      "valid": true,
      "cancellationTime": null,
      "cancellationUser": null,
      "cancellationUserId": null,
      "cancellationTimeUTC": null,
      "firstName": "Beatrice",
      "lastName": "Henry",
      "sex": "F",
      "birthDate": 95385600000,
      "uniqueIdentifier": "PAT2",
      "phone": "(545)-131-9249",
      "mobilePhone": "(545)-131-9249",
      "email": "beatrice.henry27@example.com",
      "language": "es_ES",
      "deathDate": null,
      "isDead": null,
      "patientId": "444",
      "accountNumber": null
    }
  },
  .... {
  ....
}
]

```


Chiamata del servizio all'interno della form

Questo servizio può essere richiamato anche all'interno della form, utilizzando nell'editor JavaScript la funzione presente in libreria e chiamata **FIND CONCEPT INSTANCEVALUE**:

```
var findFMConceptInstanceValue = function() {  
    CustomAPIService.findFMConceptInstanceValue("conceptName", "templateName", "templateVersion", "Date :  
from", "Date : to", "patientId", "includeDraft").then(  
    function successCallback(data) {  
        console.log(data);  
    },  
    function errorCallback(error) {  
        console.log(error);  
    });  
};
```

Questa funzione consente di ottenere lo stesso risultato sopra illustrato e utilizza gli stessi parametri di input a eccezione del token.

14. WS PER RECUPERARE VALORI DEI CONCETTI E ID DELLE ISTANZE TRAMITE PATIENT ID

Il webservice (REST) che permettere di recuperare i valori dei concetti e i rispettivi id delle istanze delle form tramite id del paziente è **http://[host]:[port]/HIS/rest/form_manager/FMConcept/findFMConceptValueByPatientIdExt** (ogni installazione è caratterizzata da un host e una port specifici).

NOTA: il servizio restituisce solo i valori afferenti alle **istanze** delle form in stato **PUBLISHED** a meno di non settare in chiamata il parametro **includeDraft** a true.

Metodo: GET

Parametri dell'header:

- **token** (String): **OBBLIGATORIO**. il token ottenuto dal WS di tokenissue suddetto.
- **patientId** (String): **OBBLIGATORIO**. Id del paziente sul quale si vogliono recuperare i concetti. E' lo stesso patientId passato nell'oggetto patient nella chiamata al servizio urlByCodeAndVersionExt.
- **from** (String): data dalla quale si vogliono recuperare i concetti (da esprimere nel formato YYYY-MM-DD)
- **to** (String): data fino alla quale si vogliono recuperare i concetti (da esprimere nel formato YYYY-MM-DD)
- **includeDraft** (Boolean): se true, nei risultati include anche i valori delle istanze salvate in bozza

NOTE:

E' possibile specificare uno solo dei parametri **from** o **to** per filtrare sulle date (**filtro inclusivo** sul giorno). Se cerco solo per data recupererò tutte le istanze compilate nel range di date, riferite a tutti i template e a tutti i pazienti.

Output: il servizio restituisce un **array di oggetti json** dove ogni oggetto risulta così composto:

- **concept:** codice del concetto ricercato
- **value:** valore del concetto nell'istanza della form corrispondente

- **date:** timestamp della data in cui l'istanza è stata salvata
- **instanceValueId:** id dell'istanza della form a cui il concetto fa riferimento
- **code:** codice del template a cui fa riferimento l'istanza.
- **version:** versione del template a cui fa riferimento l'istanza.
- **status:** stato dell'istanza (draft, published)

Errori gestiti:

- **AccessDeniedException.** Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **javax.validation.ConstraintViolationException: 1 constraint violation(s) occurred during method validation.** Constructor or Method: **public java.util.List.** Generato in caso token o patientId mancante (parametri obbligatori).
- **Error during find FMConceptValue, cannot find any template with code xxx and version yy.** Generato in caso parametro specificato in maniera errata (ad esempio versione o template non esistenti).
- **Error during find FMConceptValue, fromDate is note before toDate.** Generato in caso di data specificata nel from successiva alla data specificata in to.

Esempio di chiamata

URL: http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMConcept/findFMConceptValueByPatientIdExt

Parametri in input (HEADER):

- **token:**
eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZJZqYA.k3t9u3ckIkYr-vCkMTcX3wZhl_NxXc2T_RskLKVoNGYUyMM0Wa-
hibMwnPfkLj6MKpXjKbwidWgZnjPC8gN1gP7d3JnCBGe5Fbkn-
hdYNKqYy7udltOqFeukyzBqVOOMNI3hXliGMKSnoKiFsv179jHVIK3JoXOjeAvMsmZAPMY.vxiAgAimZrJurMVm9MgFuQ
- **patientId:** 444
- **from:** 2018-05-10
- **to:** 2018-05-11
- **includeDraft:** false

Esempio di risposta

```
[{
  "concept": "peso",
  "value": "44",
  "date": 1524840944281,
  "instanceValueId": "5ae339f088506e4b865be21e",
  "code": "test_concept",
  "version": 1,
  "status": "draft"
},
{
  "concept": "altezza",
  "value": "23",
```

```

    "date": 1524840944281,
    "instanceValueId": "5ae339f088506e4b865be21e",
    "code": "test_concept",
    "version": 1,
    "status": "draft"
  },
  {
    "concept": "peso",
    "value": "23",
    "date": 1525851745857,
    "instanceValueId": "5af2a66188506e40cb5116ec",
    "code": "test_concept",
    "version": 1,
    "status": "draft"
  },
  {
    "concept": "test.numero1a",
    "value": "43",
    "date": 1525851745857,
    "instanceValueId": "5af2a66188506e40cb5116ec",
    "code": "test_concept",
    "version": 1,
    "status": "draft"
  }
}]

```

Chiamata del servizio all'interno della form

Questo servizio può essere richiamato anche all'**interno della form**, utilizzando nell'editor JavaScript la funzione presente in libreria e chiamata **FIND CONCEPTS BY PATIENT**:

```

var findFMConceptValueByPatientId = function() {
  CustomAPIService.findFMConceptValueByPatientId("Date : from", "Date : to", "patientId", "includeDraft").then(
    function successCallback(data) {
      console.log(data);
    },
    function errorCallback(error) {
      console.log(error);
    }
  );
};

```

Questa funzione consente di ottenere lo stesso risultato sopra illustrato e utilizza gli stessi parametri di input a eccezione del token.

15. WS PER RECUPERARE ULTIMI VALORI DEI CONCETTI RILEVATI

Il webservice (REST) che permettere di recuperare gli **ultimi valori dei concetti rilevati** e le rispettive istanze delle form associate (i formInstanceId) tramite codice del concetto è **http://[host]:[port]/HIS/rest/form_manager/FMConcept/getLastFMConceptValueExt** (ogni installazione ha host e port specifici).

NOTA: il servizio restituisce solo i valori afferenti alle **istanze** delle form in stato **PUBLISHED** a meno di non settare in chiamata il parametro includeDraft a true.

Metodo: GET

Parametri dell'header:

- **token** (String): **OBBLIGATORIO**. il token ottenuto dal WS di tokenissue suddetto.
- **concept** (String): **OBBLIGATORIO**. Codice del concetto che si vuole ricercare. E' possibile ricercare anche per **concetto padre** con una sintassi del tipo patient.* dove patient è il concetto padre.
- **templateName** (String): nome del template della form dove si vuole ricercare il concetto. Se cerco solo per templateName recupererò tutte le istanze di quel template, riferite a tutti i pazienti e a tutte le date.
- **version** (Long): versione del template della form dove si vuole ricercare il concetto. Da specificare solo nel caso in cui sia specificato anche il parametro templateName. Nel caso in cui sia specificato il templateName senza version, sarà restituita l'**ultima versione** del template.
- **patientId** (String): id del paziente sul quale si vogliono recuperare i concetti. Se cerco solo per patientId recupererò tutte le istanze compilate su quel paziente, riferite a tutti i template e a tutte le date.
- **includeDraft** (Boolean): se true, nei risultati include anche i valori delle istanze salvate in bozza

NOTE:

E' obbligatorio specificare **almeno un parametro** tra templateName e patientId per poter effettuare la ricerca.

Output: il servizio restituisce un **array di oggetti json** dove ogni oggetto risulta così composto:

- **concept:** codice del concetto ricercato
- **value:** valore del concetto nell'istanza della form corrispondente
- **date:** timestamp della data in cui l'istanza è stata salvataggio
- **instanceValueId:** id dell'istanza della form a cui il concetto fa riferimento
- **code:** codice del template a cui fa riferimento l'istanza. Valorizzato solo nel caso in cui il servizio sia chiamato con il parametro di header templateName
- **version:** versione del template a cui fa riferimento l'istanza. Valorizzato solo nel caso in cui il servizio sia chiamato con il parametro di header templateName
- **status:** stato dell'istanza (draft, published)

Errori gestiti:

- **AccessDeniedException.** Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **javax.validation.ConstraintViolationException: 1 constraint violation(s) occurred during method validation.** Constructor or Method: **public java.util.List**. Generato in caso token o concept mancante (parametri obbligatori).
- **Error during find FMConceptValue , need to specify at least 1 parameter between template or patient id.** Generato in caso di mancata definizione di almeno un parametro tra templateName, patientId.

- **Error during find FMConceptValue, cannot find any template with code xxx and version yy.** Generato in caso parametro specificato in maniera errata (ad esempio versione o template non esistenti).
- **Error during find FMConceptValue, cannot specify a version without a template name.** Generato in caso di parametro version specificato senza parametro template.

Esempio di chiamata

URL: http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMConcept/getLastFMConceptValueExt

Parametri in input (HEADER):

- **token:**
eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH
4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZJZqYA.k3t9u3ckIkYr-vCkMTcX3wZhI_NxXc2T_RskLKVoNGYUyMM0Wa-
hibMwnPfkLj6MKpXjKbwidWgZnjPC8gN1gP7d3JnCBGe5Fbkn-
hdYNNKqYy7udltOqFeukyzBqVOOMNI3hXliGMKSnOKiFsv179jHVIK3JoXOjeAvMsmZAPMY.vxiAgAimZrJurMVm9MgF
uQ
- **concept:** peso
- **templateName:** test_concept_peso
- **version:** 1
- **patientId:** 444
- **includeDraft:** false

Esempio di risposta

```
[{
  "concept": "peso",
  "value": "4",
  "date": "1526028961976",
  "instanceValueId": "5af55aa188506e40cb0cf800",
  "code": "test_concept_summary_2",
  "version": 2,
  "status": "draft"
}]
```

Chiamata del servizio all'interno della form

Questo servizio può essere richiamato anche all'interno della form, utilizzando nell'editor JavaScript la funzione presente in libreria e chiamata **FIND CONCEPTS BY PATIENT**:

```
var getLastConcept = function() {
  CustomAPIService.getLastFMConceptValue("conceptName", "templateName", "templateVersion", "patientId", "includeDraft").then(
    function successCallback(data) {
      console.log(data);
    },
    function errorCallback(error) {
      console.log(error);
    }
  );
};
```

Questa funzione consente di ottenere lo stesso risultato sopra illustrato e utilizza gli stessi parametri di input a eccezione del token.

16. WS PER RECUPERARE LISTA DEI CONCETTI DEFINITI SU INSTALLAZIONE

Il webservice (REST) che permettere di recuperare tutti i concetti definiti sulla singola installazione di Form Manager 3 e i corrispondenti valori associati è **http://[host]:[port]/HIS/rest/form_manager/FMConcept/findAllFMConceptsExt** (ogni installazione è caratterizzata da un host e una port specifici). Il servizio restituisce sia i concetti validi che quelli invalidati.

Metodo: GET

Parametri dell'header:

- **token** (String): **OBBLIGATORIO**. il token ottenuto dal WS di tokenissue suddetto

Output: il servizio restituisce un **array di oggetti json** dove ogni oggetto risulta così composto:

- **id** (Long): identificativo del concetto
- **insertUserId** (Long): id dell'utente che ha inserito il concetto
- **insertUser** (String): username dell'utente che ha inserito il concetto
- **insertTimeUTC** (Date): data di inserimento del concetto senza timezone
- **insertTime** (Date): data di inserimento del concetto
- **updateUserId** (Long): id dell'utente che ha aggiornato il concetto
- **updateUser** (String): username dell'utente che ha aggiornato il concetto
- **updateTimeUTC** (Date): data di aggiornamento del concetto senza timezone
- **updateTime** (Date): data di aggiornamento del concetto
- **organizationId** (Long): id relativo all'installazione del Form Manager 3
- **valid** (Boolean): false se il concetto è stato cancellato e quindi non è più valido
- **cancellationTime** (Date): data di cancellazione dell'istanza
- **cancellationUser** (String): username dell'utente che ha cancellato l'istanza
- **cancellationUserId** (Long): id dell'utente che ha cancellato l'istanza
- **cancellationTimeUTC** (Date): data di cancellazione dell'istanza senza timezone
- **code** (String): codice del concetto come definito nel configuratore dei concetti
- **description** (oggetto): json contenente la lista delle descrizioni, con descrizione definita per ogni lingua

Errori gestiti:

- **AccessDeniedException**. Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.

Esempio di chiamata

URL: *http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMConcept/findAllFMConceptsExt*

Parametri in input (HEADER):

- **token:**
eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZIZqYA.k3t9u3ckIkYr-vCKMTcX3wZhI_NxXc2T_RskLKVoNGYUyMM0Wa-hdYnKqYy7udltOqFeukyzBqVOOMNI3hXliGSnOKIFsv179jHVIK3JoXOjeAvMsmZAPMY.vxiAgAimZrJurMVm9MgFuQ

Esempio di risposta

```
[ {  
  "id": "5ae3240088506e4b865be203",  
  "insertUserId": null,  
  "insertUser": null,  
  "insertTimeUTC": null,  
  "insertTime": null,  
  "updateUserId": 2,  
  "updateUser": "Gianpaolo Zanella",  
  "updateTimeUTC": 1524836372772,  
  "updateTime": 1524836372772,  
  "organizationId": null,  
  "valid": true,  
  "cancellationTime": null,  
  "cancellationUser": null,  
  "cancellationUserId": null,  
  "cancellationTimeUTC": null,  
  "code": "TEST",  
  "description": {  
    "it_IT": "test_ita",  
    "en_GB": "test_en"  
  }  
},  
...]
```

17. WS PER RECUPERARE LISTA DEI CONCETTI ESPOSTI DA UN TEMPLATE

Il webservices (REST) che permettere di recuperare tutti i **concetti definiti** su un **template** di Form Manager 3 è **http://[host]:[port]/HIS/rest/form_manager/FMConcept/FindFmConceptFromTemplateExt** (ogni installazione è caratterizzata da un host e una port specifici). Il servizio restituisce i concetti esposti dai template sia pubblicati che in bozza.

Metodo: GET

Parametri dell'header:

- **token** (String): **OBBLIGATORIO**. Il token ottenuto dal WS di tokenissue suddetto
- **code** (String): **OBBLIGATORIO**. Il codice del template sul quale si vogliono ricercare i concetti.
- **version** (Long): versione del template della form dove si vuole ricercare il concetto. Nel caso in cui il parametro non sia passato come input, sarà restituita l'**ultima versione** del template.

Output: il servizio restituisce un **array di oggetti json** dove ogni oggetto risulta così composto:

- **concept** (String): codice del concetto ricercato
- **value**: sempre a null per questo servizio
- **date**: sempre a null per questo servizio
- **instanceValueId**: sempre a null per questo servizio
- **code**: codice del template
- **version**: versione del template (ultima versione in caso di parametro di input version non specificato)
- **status**: sempre a null per questo servizio
- **bundleValues**: json contenente la lista delle label definite sul concetto, con label definita per ogni lingua

Errori gestiti:

- **AccessDeniedException.** Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **it.gpi.his.commons.exceptions.generics.ApplicationErrorException: Error during find FMConceptValue, template does not exists.** Generato in caso di parametro code non passato o in caso di code/version non esistente.

Esempio di chiamata

URL: http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMConcept/FindFmConceptFromTemplateExt

Parametri in input (HEADER):

- **token:**
`eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH`
- **code:** `test_concept_ws`
- **version:** `1`

Esempio di risposta

```
[{
  "concept": "gait",
  "value": null,
  "date": null,
  "instanceValueId": null,
  "code": "test_concept_ws",
  "version": 1,
  "status": null,
  "bundleValues": {
    "it_IT": "Andatura",
    "en_GB": "Gait"
  }
},
{
  "concept": "balance",
  "value": null,
  "date": null,
  "instanceValueId": null,
  "code": "test_concept_ws",
  "version": 1,
  "status": null,
  "bundleValues": {
    "it_IT": "Equilibrio",
    "en_GB": "Balance"
  }
},
{
  "concept": "score",
  "value": null,
  "date": null,
  "instanceValueId": null,
  "code": "test_concept_ws",
  "version": 1,
  "status": null,
  "bundleValues": {
    "it_IT": "Punteggio",
    "en_GB": "Score"
  }
}]
```


18. SUMMARY FIELDS

Nella definizione di una form nel configuratore è possibile definire alcuni campi come **summary**. I summary fields sono da intendersi come informazioni di riepilogo relative alla form (es: punteggi parziali, informazioni peculiari ecc...).

Utilizzo dei summary nella form

Per definire un **campo di input** come summary occorre associare il tag summary come nell'esempio seguente

```
<input type="text" summary="caduta3mesi" ng-model="formData.caduta3Mesi">
```

Il campo summary dovrà contenere una **label** definita nell'**Editor Bundles**, come nell'immagine seguente:

Lingua	Chiave	Valore
it_IT	caduta3mesi	Caduta ultimi 3 mesi

- In caso di **label non definita** nell'Editor Bundles, l'Editor HTML segnalerà il seguente errore **Summary attribute refers to a not existing bundle key** e il sistema non consentirà il salvataggio della form (bottone disabilitato).
- Se il tag summary è associato a un **campo diverso da un input** sarà segnalato il seguente warning: **Summary attribute is not on input and will be ignored** e sarà ignorato.

```
21 <label>Temperatura corporea</label>
22 <input type="text" concept="temperature" ng-model=
23 </div>
26 <label summary>Dolore</label>
27 <input type="text" summary="pain" ng-model="formDa
28 </div>
```

E' inoltre possibile definire un summary associato a un **concetto** come nell'esempio seguente. In questo caso la label del summary sarà ereditata dalla label del concetto.

```
<input type="text" concept="peso" summary ng-model="formData.patient.weight">
```

A destra della pagina è visibile un **riepilogo dei summary** definiti per la form:

☒ Mostra JSON dei summary fields

Mostra JSON dei summary fields	
pain	
it_IT	dolore
peso	
it_IT	peso

```
{ "pain": { "it_IT": "dolore" }, "peso": { "it_IT": "peso" } }
```

Recupero dei summary nei servizi chiamati da applicativi esterni

Nella chiamata ai servizi `urlByIdExt` e `urlByCodeAndVersionExt`, i summary fields saranno associati al `formDocument` con un array di oggetti json chiamato **summaryFieldsData** (vedi esempio sotto). L'array L'oggetto conterrà i summary definiti per la form. Per ognuno saranno riportati la chiave (**key**), il valore (**value**), i **bundles** associati per le diverse lingue e un boolean, **isConcept**, che indicherà se quel summary è un concetto (true) o no (false).

```
"formDocument": {
  "initData": { ...
},
"summaryFieldsData": [
  {
    "key": "peso",
    "value": "4",
    "isConcept": true,
    "bundles": { "it_IT": "peso" }
  },
  {
    "key": "pain",
    "value": "no",
    "isConcept": false,
    "bundles": { "it_IT": "dolore" }
  }
]
```

19. WS PER RECUPERARE I SUMMARY FIELDS TRAMITE ISTANZA

Il webservice (REST) che permettere di recuperare i summary fields di una o più istanze è **http://[host]:[port]/HIS/rest/form_manager/FMSummaryFields/findSummaryFieldsByInstanceExt** (ogni installazione è caratterizzata da un host e una port specifici).

Metodo: PUT

Parametri dell'header:

- **token** (String): **OBBLIGATORIO**. il token ottenuto dal WS di tokenissue suddetto.
- **formInstanceId** (String): **OBBLIGATORIO**. id dell'istanza di cui si vogliono recuperare i summary fields.
- **version** (Long): versione dell'istaza. Da specificare solo nel caso in cui sia specificato anche il parametro formInstanceId. Nel caso in cui sia specificato il formInstanceId senza version, sarà restituita l'**ultima versione** dell'istanza.

Parametri del body:

- **fmlInstanceList** (array): array di formInstanceId appartenenti alle istanze di cui si vogliono recuperare i summaryFields.

Output: il servizio restituisce un **array dei summaryFields** relativi alle istanze specificate.

Errori gestiti:

- **AccessDeniedException**. Generato in caso di token non generato correttamente. Cause = INVALID_TOKEN.
- **javax.validation.ConstraintViolationException: 1 constraint violation(s) occurred during method validation.\nConstructor or Method: public java.util.List.** Generato in caso token mancante (parametri obbligatori).
- **Error during find FMConceptValue ,cannot specify a version for a fmlInstanceList.** Generato in caso di parametro version specificato quando è settato il parametro fmlInstanceList.
- **Error during find FormInstanceDto by instanceId, cannot find FMInstanceDto with instanceId: xxxx.** Generato in caso di formInstanceId passato non esistente.

Esempio di chiamata

URL: `http://devsrv03.erossi.org:8080/HIS/rest/form_manager/FMSummaryFields/findSummaryFieldsByInstanceExt`

Parametri in input (HEADER):

- **token:**
`eyJhbGciOiJBMTI4S1ciLCJlbmMiOiJBMTI4Q0JDLUhTMjU2In0.ptkhKv3rLMF4BtZg6Wx_4AivHIRrf_Qq8oW6ZbsKOH4qHEB1Lr-Ndw.4cS1x8YstZ87PmjsZIZqYA.k3t9u3ckIkYr-vCkMTcX3wZhI_NxXc2T_RskLKVoNGY`
- **formInstanceId:** `5af55aa188506e40cb0cf800`
- **version:** `1`

Parametri del body:

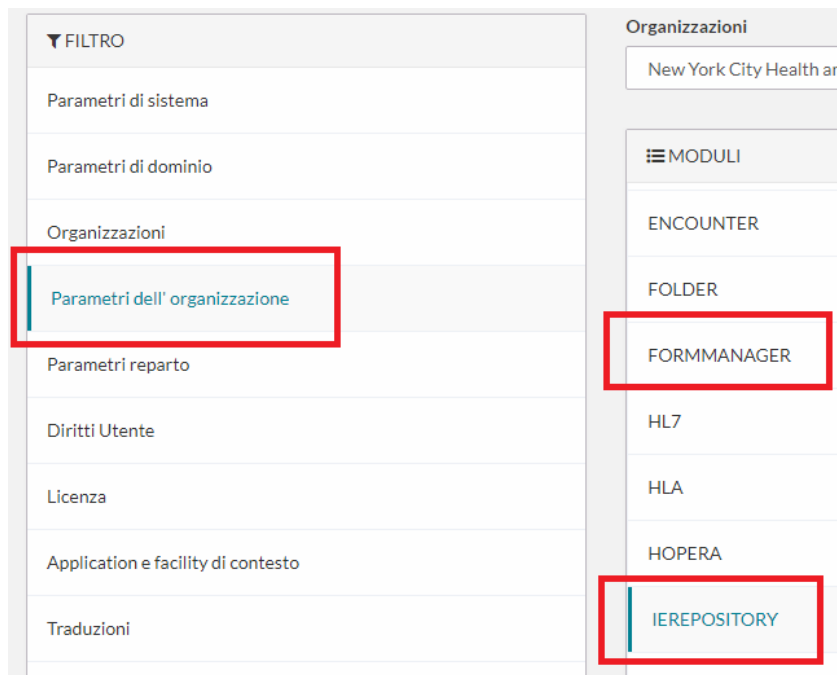
- `["5b64329b88506e5be89668a7", "5b6434e788506e5be89668ae"]`

Esempio di risposta (caso in cui si passa nel body la fmlInstanceList)

```
[
  {
    "summaryFields": [
      {
        "key": "sum",
        "value": "2",
        "isConcept": false
      },
      {
        "key": "total",
        "value": "4",
        "isConcept": false
      }
    ],
    "fmlInstanceId": "5b64329b88506e5be89668a7",
    "version": 3
  },
  {
    "summaryFields": [
      {
        "key": "sum",
        "value": "9",
        "isConcept": false
      },
      {
        "key": "total",
        "value": "9",
        "isConcept": false
      }
    ],
    "fmlInstanceId": "5b6434e788506e5be89668ae",
    "version": 1
  },
  {
    "summaryFields": [
      {
        "key": "sum",
        "value": "0",
        "isConcept": false
      },
      {
        "key": "total",
        "value": "0",
        "isConcept": false
      }
    ],
    "fmlInstanceId": "5b64351b88506e5be89668b0",
    "version": 1
  }
]
```

20. INTEGRAZIONE CON REPOSITORY

Per attivare l'integrazione di Form Manager 3 con il repository installato presso il cliente, occorre prima configurare nella sezione **Admin** di NGH i **parametri dell'organizzazione** relativi a **FORM MANAGER** e **IE-REPOSITORY** (menu abilitato al solo utente Amministratore). Per scopi di test è stato utilizzato per FM3 l'ambiente NGH e il repository di test del server **tstcss01** (<http://tstcss01.gpi.it:8080/HIS>; utenza: **adminfm3/adminfm3**) di cui si riportano le rispettive configurazioni.



Parametri di configurazione di FORMMANAGER

I parametri sono estraibili dal file di configurazione del repository **xsd.properties**.

- **Attivazione invio repository:** **true** per attivare l'integrazione. Se impostato a false, i servizi REST suddetti restituiranno l'errore **Organization parameter FORMMANAGER.FORMMANAGER_REPOSITORY_ENABLED not enabled** in caso di form con **flag Invio al repository abilitato** perché l'invio risulterà disabilitato a livello di organizzazione.

Attivazione invio repository	true
------------------------------	------

Parametri di configurazione di IEREPOSITORY

I parametri elencati sono estraibili dai file di configurazione del repository **xsd.properties** e **repository.properties**.

- **IERepository – TokenIssue (Sistema, Password, Audience):** parametri per l'autenticazione dipendenti dalla specifica installazione, qui configurati per l'ambiente di test **tstcss01**. I dati di configurazione sono presenti nel file. **Sistema = IE-Repository, Password= password, Audience=IE-Repository.**
- **IERepository – Url servizi rest:** url da utilizzare per l'invio dei documenti al repository. Per **tstcss01** pari a <http://tstcss01:8080/DocumentRepository/servicesAccessGate/repositoryAccessGateway>

LISTA DI PARAMETRI	
IERepository - Modalità di apertura documenti di tipo text/url: [NONE, HIDDEN, WRAPPED]	false
IERepository - TokenIssue Sistema	IE-Repository
IERepository - TokenIssue Password	password
IERepository - TokenIssue Audience	IE-Repository-cat
IERepository - Mostra selezione Auditable Event se non è definito un default	false
IERepository - Url servizi rest	http://tstcss01:8080/DocumentRepository/servicesAccessGateway/repositoryAccessGateway
IERepository - Nome EJB Remote	ejb:DocumentRepository/EjbDocumentRepository//ClientConsumerBean!com.gpi.repository.client.services.interfaces.ClientConsumerRemoteInterface

Parametri di configurazione della singola form

I parametri configurabili per singola form sono da settare in **Opzioni template** come indicato in figura.

Opzioni template - FM3 - FM3 - v.1

Invio al Repository abilitato <input type="checkbox"/>	CDA2 abilitato <input type="checkbox"/>	Invia txt file <input type="checkbox"/>	Annullamento abilitato <input type="checkbox"/>
Firma digitale abilitata <input type="checkbox"/>			

- **Invio al repository abilitato:** abilita l'invio al repository per la singola form.
 - Se il flag è abilitato, quando l'utente salverà la form in Definitivo automaticamente il documento sarà inviato al repository.
 - In caso di errore nell'invio al repository, il salvataggio in Definitivo sarà interrotto.
 - In caso di documento già pubblicato e quindi già inviato al repository, non sarà possibile la ripubblicazione del documento (bottone Definitivo disabilitato)
 - In caso di documento già pubblicato, cliccando su Stampa, il PDF sarà recuperato dal repository.
 - In caso di utilizzo dei servizi REST suddetti senza oggetto patient in input, i servizi restituiranno il seguente errore: **Patient is not consistent with repository: could not continue**
- **CDA2 abilitato:** se abilitato la form supporterà il formato CDA2 quindi sarà possibile inviare al repository l'xml corredato dell'allegato in pdf del documento. L'xml dovrà essere definito nell'apposita sezione **Editor XML** presente in Opzioni template che risulterà visibile alla selezione del flag CDA2 come mostrato in figura. Questa op-

zione sarà attivabile solo se è stata prima selezionata l'opzione **Invio al repository abilitato**. Un esempio di xml

```

EditorXML
1 <xsl:stylesheet version="1.1" xmlns="urn:h17-org:v3"
2     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3     xmlns:xs="http://www.w3.org/2001/XMLSchema"
4     xmlns:java="http://xml.apache.org/xalan/java"
5     xmlns:dyn="http://exslt.org/dynamic"
6     exclude-result-prefixes="xs java dyn">
7
8     <xsl:output method="xml" omit-xml-declaration="no" indent="yes" />
9
10    <xsl:variable name="PATIENT_KEY_EXTERNAL_IDENTIFIER" select="'ASL FOGGIA'" />
11
12    <xsl:variable name="OID_ROOT_CDA" select="'2.16.840.1.113883.2.9.3.33.1508413597'" />
13    <xsl:variable name="OID_NGH" select="concat($OID_ROOT_CDA, '.1')" />
14    <xsl:variable name="OID_REPORT_PLAN" select="concat($OID_NGH, '.1')" />
15

```

- **Invia txt file:** se abilitato sarà possibile inviare al repository un file di testo costruito a partire dai dati inseriti nella form. Per estrarre i dati da inviare nel file txt occorre utilizzare una funzione JavaScript, come ad esempio quella definita nella libreria JavaScript chiamata **REGISTER EXTRACT SUMMARY FIELDS** e mostrata in figura. Nel corpo della funzione va inserita la logica per estrarre i dati di interesse che vanno a popolare la variabile txt.

Nome	Descrizione
Register Extract Summary	This function is used to create the Summary Text document

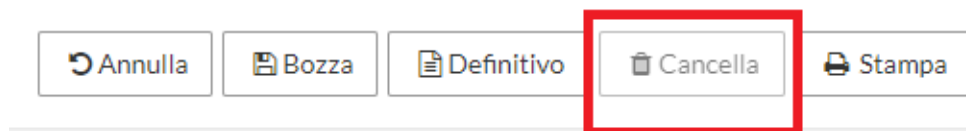
```

1 var extractSummaryText = function(instanceValueId) {
2   var txt = "summary txt to send to repository";
3   return txt;
4 };
5 registerExtractSummaryTextFunction(extractSummaryText);

```

Questa opzione sarà attivabile solo se è stata prima selezionata l'opzione **Invio al repository abilitato**.

- **Annullamento abilitato:** se abilitato sarà possibile annullare il documento già inviato sul repository. Se abilitata l'opzione, a runtime comparirà un **bottone Cancella** disabilitato. **Cliccando su Cancella il documento sarà annullato sul repository e lo stato dell'istanza passerà in stato DELETED.** Il bottone Cancella risulterà **abilitato** solo in caso di **form** già **pubblicata** (e quindi già inviata al repository), poiché l'annullamento è previsto solo in caso di invio al repository già avvenuto. Il bottone risulterà **disabilitato** anche in caso di **readOnly=true** settato nelle runtimeOptions. **Questa opzione è attivabile solo se è stata prima selezionata l'opzione **Invio al repository abilitato**.**



- **Firma digitale abilitata:** se abilitato sarà possibile firmare digitalmente il documento da inviare al repository.

Metadati

L'invio al repository del documento generato a partire dalla form richiede la registrazione di alcuni **metadati**, con **nomi** e **obbligatorietà** diversa a seconda del repository utilizzato. Per registrare i metadati occorre utilizzare la funzione JavaScript presente anche in libreria chiamata REGISTER METADATA così definita: **registerMetaData("metaDataName","value","isMultiple")**. Il flag isMultiple è da settare a true per i metadati che possono avere valori multipli.

Un esempio di metadato che in genere occorre registrare è **objectType** che può assumere i seguenti valori: 1:EVENTO, 2: DOCUMENTO, 3: EXTERNAL LINK, 4: MEDICAL RECORD, 5: CLINICAL FOLDER, 6: CDA, 9:CDA2, 10: CDA2 LABORATORIO. In questo caso, nell'Editor JavaScript della form, il metadato sarà così registrato: **registerMetaData("objectType", "9", "false")**.

Se i metadati non sono correttamente definiti, l'invio al repository genererà un errore e il salvataggio in Definitivo dell'istanza sarà interrotto.

In caso di metadati definiti, nel json che rappresenta l'istanza della form, vi sarà un array di oggetti JSON così rappresentati:

```
"metaData":  
[  
  {  
    "key": "xdsbClassCode",  
    "value": "REF"  
  },  
  { ... }  
]
```