Pannelli griglie e alberi

Indice generale

Pannelli griglie e alberi

Costruisci velocemente applicazioni web e mobile di carattere gestionale con i widget Pannello e Albero di Instant Developer Cloud.

Introduzione

La maggior parte delle applicazioni cloud moderne, sia di carattere gestionale che di trasformazione digitale, devono presentare e consentire di modificare i dati che l'applicazione sta gestendo.

Per questo scopo, è largamente accettato il paradigma UX *Griglia / Dettaglio*, in cui i dati disponibili vengono mostrati tramite un elemento visuale di tipo griglia con funzionalità di ricerca, selezione e raggruppamento. Agendo opportunamente su una riga della griglia, viene visualizzato un dettaglio del dato selezionato e a questo punto è possibile anche modificarlo.

Tale paradigma vale anche in caso di applicazioni mobile, facendo le opportune semplificazioni dovute al tipo di interazione (tocco invece che mouse) e alla più piccola dimensione dello schermo.

Per poter soddisfare queste necessità, Instant Developer Cloud mette a disposizione un nuovo insieme di widget denominato *IdfWidget* che permettono di costruire con il massimo della velocità front-end dedicati alla gestione dei dati.

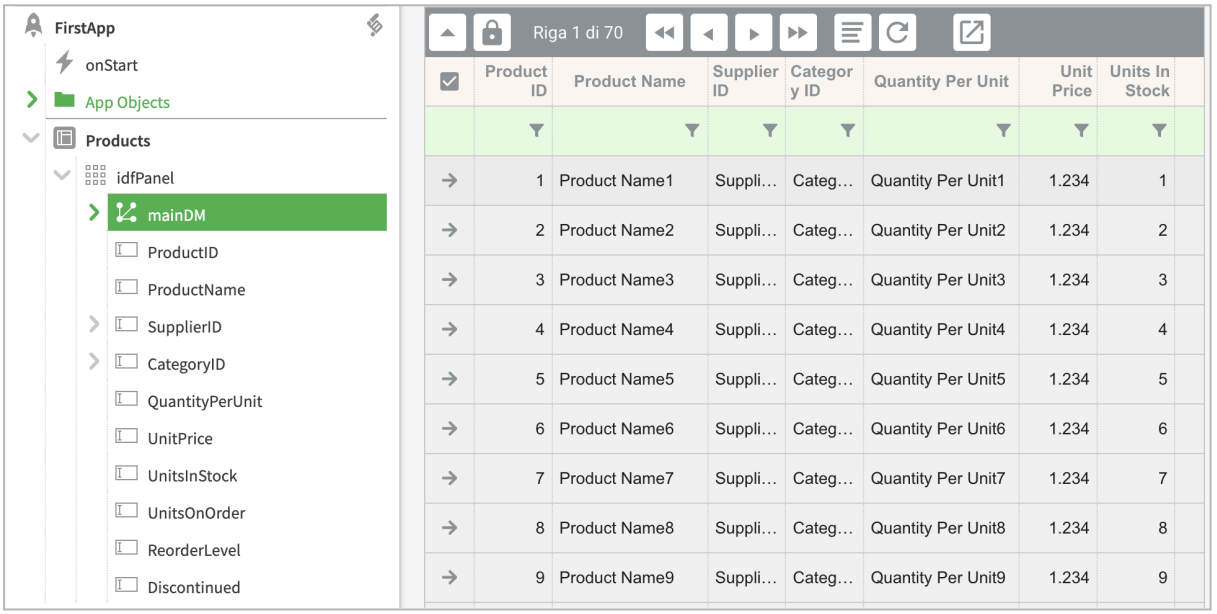
Tali widget, inoltre, si integrano senza soluzione di continuità con tutti gli altri elementi visuali disponibili nella piattaforma, permettendo così di poter continuare a progettare le interfacce con il massimo della flessibilità e contemporaneamente poter assolvere ai compiti di presentazione e gestione dati con il massimo della semplicità e velocità.

La prima parte del nome *IdfWidget* deriva dal fatto che i componenti resi disponibili sono i medesimi presenti nella versione *Foundation* di Instant Developer (IDF). Chi ha esperienza con questo tipo di componenti, sarà ancora più facilitato nell'utilizzo dei widget, anche se è consigliabile comunque leggere questo manuale per poter cogliere tutte le novità di funzionamento disponibili solo nella versione *Cloud*.

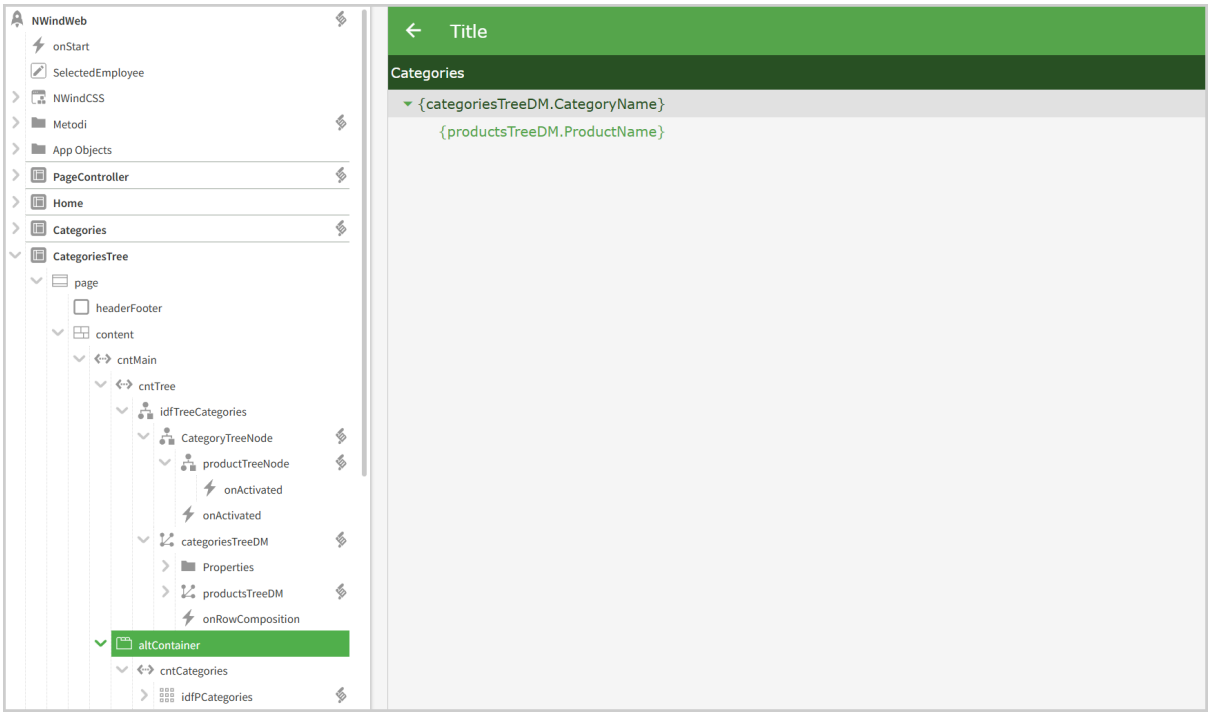
Widget disponibili

- *IdfPanel*: rappresenta l'intero pannello, comprensivo di lista (griglia) e dettaglio.
- *IdfField*: rappresenta una colonna della griglia e un campo del dettaglio.
- *IdfGroup*: rappresenta un gruppo di campi che vengono presentati in maniera coerente.
- *IdfPage*: rappresenta un pagina di campi quando il pannello è mostrato in modalità multipagina. Viene usato quando il pannello deve gestire decine di campi che vengono visualizzati in modo alternato.

- *IdfTree:* rappresenta un elemento di interfaccia utente in grado di visualizzare i dati in un albero (una struttura gerarchica).
- *IdfTreeNode:* rappresenta un nodo della visualizzazione ad albero.



Esempio di pannello per la gestione di una tabella di dati relativi ai prodotti



Esempio di albero per la visualizzazione di categorie e prodotti

Funzionalità dei pannelli

L'elenco delle funzionalità e dei comportamenti gestiti in automatico dai pannelli è veramente vasto. Qui verranno elencati solo quelli principali.

1. *Query By Example*: i pannelli permettono all'utente di inserire in modo semplice dei criteri di ricerca per selezionare i dati di interesse.

2. *Live Scrolling*: i pannelli sono predisposti per visualizzare in modalità live scrolling un qualunque numero di record, anche nell'ordine delle centinaia di migliaia. Vengono gestite sia righe ad altezza fissa che righe ad altezza variabile
3. *Legami fra pannelli e altri oggetti grafici*: tramite gli opportuni eventi a livello di datamap è molto semplice coordinare il funzionamento di più pannelli in modalità master detail o in altre funzionalità.
4. *Griglia e dettaglio*: un pannello può avere sia il layout di presentazione in lista (griglia di dati) che quello in dettaglio. L'insieme dei campi visualizzati in dettaglio può essere diverso da quello della lista.
5. *Ordinamenti*: mentre viene visualizzato il layout in lista, è possibile ordinare i dati per una o più colonne della griglia del pannello.
6. *Raggruppamenti*: attivando la visualizzazione per gruppi, le righe possono essere visualizzate in modo raggruppato anche a più livelli. È possibile inserire funzioni di totalizzazione per ogni singolo campo della lista.
7. *Esportazione in formato csv*: cliccando un solo pulsante verrà scaricato il file csv corrispondente al contenuto della griglia e lo si potrà aprire in un foglio di calcolo in locale.
8. *Modifiche ai dati*: è possibile modificare i dati sia nel layout in dettaglio che direttamente sulla lista. Il pannello gestisce anche lo stato *locked* per evitare modifiche accidentali ai dati. È presente un completo sistema di validazione e reporting degli errori a livello di singolo campo o di intero pannello. I dati modificati vengono salvati tramite i documenti in maniera automatizzata e personalizzabile.
9. *Maschere di editing*: è prevista una funzione di editing controllato e formattazione al volo diversa per campi interi, decimali, valori monetari, date, ore, stringhe. Il dato viene controllato e formattato durante la digitazione e non solo all'uscita dal campo.
10. *Uso della tastiera*: sia in lista che in dettaglio è possibile navigare fra i campi semplicemente usando la tastiera. Tutti i comandi del pannello sono attivabili con i tasti funzione.
11. *Lookup e decodifiche*: sono presenti diversi meccanismi di lookup e decodifica che consentono di visualizzare dati correlati a quelli presenti nel pannello e selezionarne facilmente altri attraverso meccanismi di tipo *intellisense*.
12. *Colonne unbound*: alcune colonne della lista possono essere scollegate dai campi delle tabelle del database per inserire nella griglia icone, pulsanti, valori o altre informazioni che non verranno salvate all'interno del database.
13. *Formattazione per cella*: tramite un opportuno evento di pannello è possibile modificare le proprietà di singole celle del pannello piuttosto che di un'intera colonna della griglia. In questo modo diventa facile esprimere le regole di formattazione condizionale dei dati.
14. *Selezione multipla*: i pannelli permettono di effettuare la selezione multipla delle righe. Alcuni comandi, come l'esportazione, la cancellazione e la duplicazione agiscono sulle righe selezionate invece che solo sulla riga attiva del pannello.
15. *Tipi di controlli*: È possibile utilizzare, oltre ad edit box e combobox, anche altri tipi di controlli come ad esempio i check-box, i radio button, immagini, e più in generale, qualunque elemento visuale presente in Instant Developer Cloud.
16. *Campi BLOB*: I pannelli gestiscono in maniera automatica l'upload ed il download del contenuto di un campo BLOB presente sul database. È possibile interagire con il pannello tramite eventi per consentire l'upload dei file anche tramite il file system del server, senza memorizzazione nel database.

17. *Fixed column*: può essere interessante fissare alcune colonne della lista e fare scorrere solo le altre, come avviene anche nei migliori fogli di calcolo.
18. *Campi raggruppati e paginati*: se è necessario mostrare molti campi è possibile raggrupparli e suddividerli in pagine. I gruppi hanno una funzione di collassamento automatica ed animata che permette il funzionamento a bande del pannello.
19. *Righe raggruppate*: I pannelli permettono di effettuare una visualizzazione raggruppata dei dati nel formato lista indicando per quali campi effettuare il raggruppamento.
20. *Personalizzazione grafica*: ogni elemento del pannello possiede il suo stile e può essere associato a classi CSS. Inoltre è possibile, sempre tramite foglio CSS, modificare l'intero set di classi di base del pannello per personalizzare l'aspetto grafico in tutta l'applicazione.
21. *Web e Mobile*: essendo basato su framework IonicUI di Instant Developer Cloud, è altamente responsive ed è quindi già predisposto sia per applicazioni web che per quelle mobile.

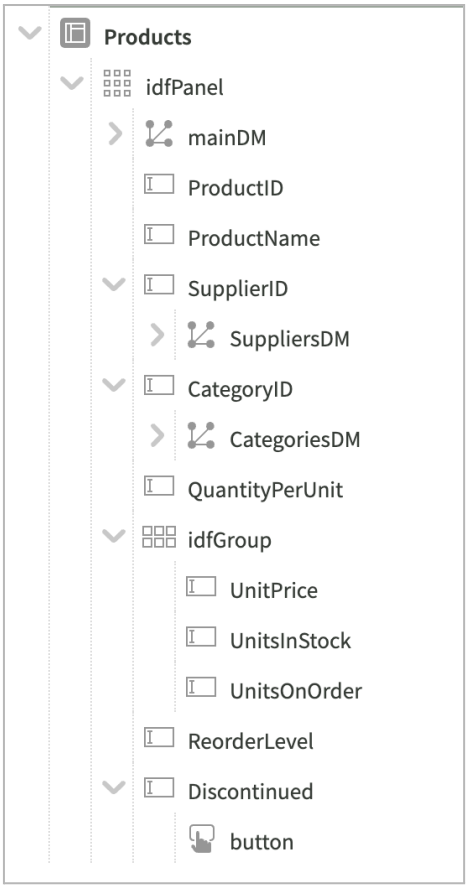
Funzionalità degli alberi

Ecco un elenco delle funzionalità e dei comportamenti gestiti in automatico dagli alberi.

1. *Semplicità d'uso:* per caricare i dati che definiscono i nodi di un albero è sufficiente utilizzare una datamap e quindi è possibile creare alberi multi livello e ricorsivi.
2. *Proprietà del nodo:* impostazione semplificata di *label* e *badge* di un nodo dell'albero.
3. *Visualizzazione compatta:* per rendere i nodi dell'albero più compatti in modo che occupino meno spazio verticale semplicemente impostando una proprietà.
4. *Selezione multipla:* gli alberi permettono di effettuare la selezione multipla dei nodi.
5. *Personalizzazione grafica*: ogni elemento del pannello possiede il suo stile e può essere associato a classi CSS. Inoltre è possibile, sempre tramite foglio CSS, modificare l'intero set di classi di base dell'albero per personalizzare l'aspetto grafico in tutta l'applicazione.
   È anche possibile inserire altri oggetti all'interno di un nodo di un albero per personalizzarne l'aspetto.
6. *Drag & drop:* utilizzo del drag & drop tra i nodi dell'albero e gestione mediante specifico evento.
7. *Uso della tastiera:* gli alberi permettono all'utente la navigazione da tastiera per spostarsi sui nodi, selezionare un nodo, espandere o contrarre un nodo.
8. *Web e Mobile*: essendo basato su framework IonicUI di Instant Developer Cloud, è altamente responsive ed è quindi già predisposto sia per applicazioni web che per quelle mobile.

Anatomia di un pannello

La struttura degli elementi visuali che compongono un pannello è mostrata nell'immagine seguente:



Il widget *IdfPanel* può essere contenuto direttamente a livello base in una videata, se non sono previsti altri elementi, oppure può essere posizionato all'interno di qualunque altro elemento di tipo container, come ad esempio *IonContent* nel caso di pagina standard Ionic.

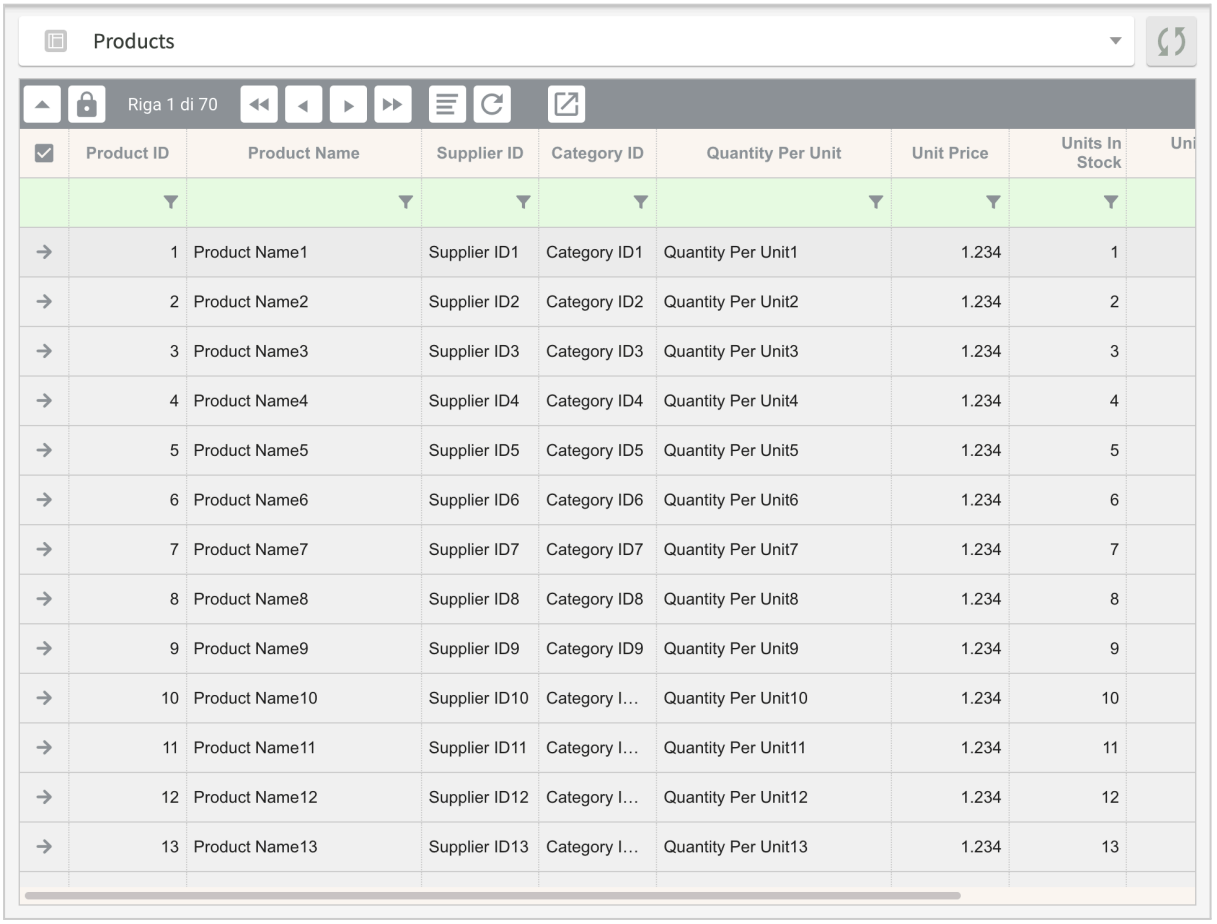
Il primo elemento figlio del pannello è la datamap principale (*mainDM*) che fornisce al pannello i dati e ne gestisce la modifica.

Sotto alla datamap troviamo i widget di tipo *IdfField* che rappresentano le colonne della griglia o i campi del dettaglio. Un elemento *IdfField* a sua volta può contenere altri elementi visuali, come si vede nell'immagine in cui l'ultimo campo (*Discontinued*) contiene un elemento di tipo *Button*. Gli elementi ulteriori possono servire per aggiungere funzionalità al campo oppure per rappresentare i dati in modo diverso, ad esempio attraverso una progress bar.

I campi possono essere raggruppati tramite un widget di tipo *IdfGroup*, che permette di gestirli in una zona specifica dell'interfaccia utente, oppure suddivisi in più pagine con un widget di tipo *idfPanelPage*, non mostrato nell'esempio.

Notiamo infine che i widget *IdfField* possono contenere anche una datamap, cosiddetta di lookup, che permette di decodificare ed effettuare la selezione del valore da altre tabelle o documenti.

Nell'editor di videate, il pannello appare come una visualizzazione in griglia, come nell'esempio seguente:



Product ID	Product Name	Supplier ID	Category ID	Quantity Per Unit	Unit Price	Units In Stock	Uni
1	Product Name1	Supplier ID1	Category ID1	Quantity Per Unit1	1.234	1	
2	Product Name2	Supplier ID2	Category ID2	Quantity Per Unit2	1.234	2	
3	Product Name3	Supplier ID3	Category ID3	Quantity Per Unit3	1.234	3	
4	Product Name4	Supplier ID4	Category ID4	Quantity Per Unit4	1.234	4	
5	Product Name5	Supplier ID5	Category ID5	Quantity Per Unit5	1.234	5	
6	Product Name6	Supplier ID6	Category ID6	Quantity Per Unit6	1.234	6	
7	Product Name7	Supplier ID7	Category ID7	Quantity Per Unit7	1.234	7	
8	Product Name8	Supplier ID8	Category ID8	Quantity Per Unit8	1.234	8	
9	Product Name9	Supplier ID9	Category ID9	Quantity Per Unit9	1.234	9	
10	Product Name10	Supplier ID10	Category I…	Quantity Per Unit10	1.234	10	
11	Product Name11	Supplier ID11	Category I…	Quantity Per Unit11	1.234	11	
12	Product Name12	Supplier ID12	Category I…	Quantity Per Unit12	1.234	12	
13	Product Name13	Supplier ID13	Category I…	Quantity Per Unit13	1.234	13	

Nella parte superiore, possiamo notare le seguenti zone:

1) Toolbar e statusbar.
2) Intestazione delle colonne.
3) Riga per l'inserimento dei filtri.
4) Righe con l'anteprima dei valori presenti nella griglia.

Sul lato sinistro, possiamo notare una colonna laterale che mostra lo stato delle righe e permette di effettuare la multi-selezione.

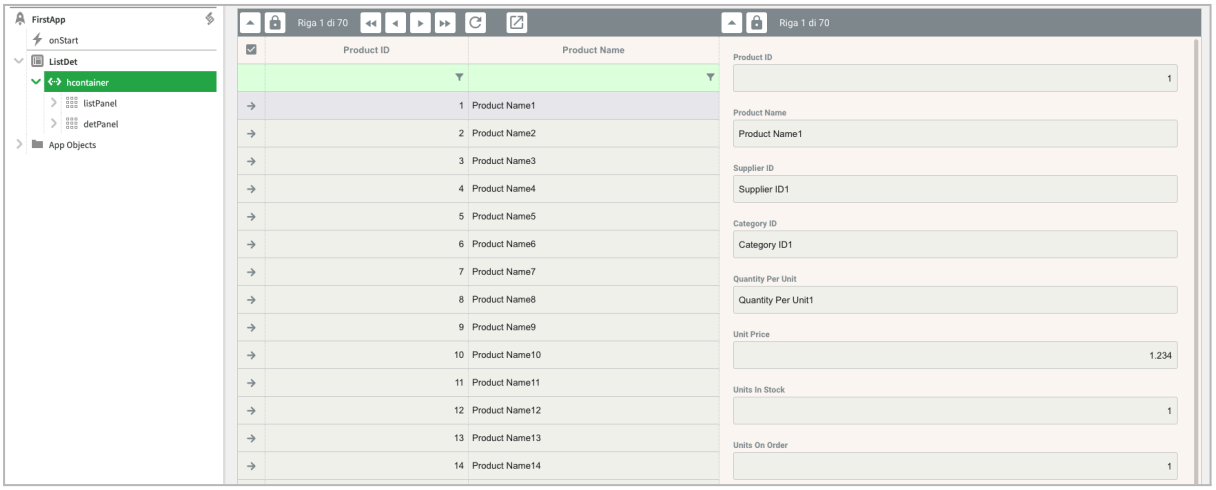
Creazione e modifica di un pannello

Creazione di un pannello

Prima di iniziare ad utilizzare i pannelli, è necessario importare il pacchetto *FluidUI*, che contiene la libreria *IDFWidgets*, con le definizioni degli elementi che compongono i pannelli.

A questo punto è possibile inserire il componente *IdfPanel*, sia a livello di videata, che a livello di elemento container. Solitamente il pannello viene inserito direttamente all'interno di una videata vuota, se questa videata serve solo per visualizzare e gestire i dati in modalità griglia e dettaglio.

Se, invece, la videata deve contenere più elementi del singolo pannello, come, ad esempio, visualizzazioni lista / dettaglio o ulteriori zone di gestione dei dati, si consiglia di inserire degli elementi *Container* con layout *horizontal* e *vertical*.



Pannelli affiancati grazie ad un container a layout orizzontale

Subito dopo aver inserito il componente *IdfPanel*, nell'albero degli oggetti viene selezionata la datamap principale del pannello. A questo punto è possibile indicare il tipo di documenti che verranno mostrati nel pannello e il sistema genererà automaticamente sia il layout in lista che quello di dettaglio.

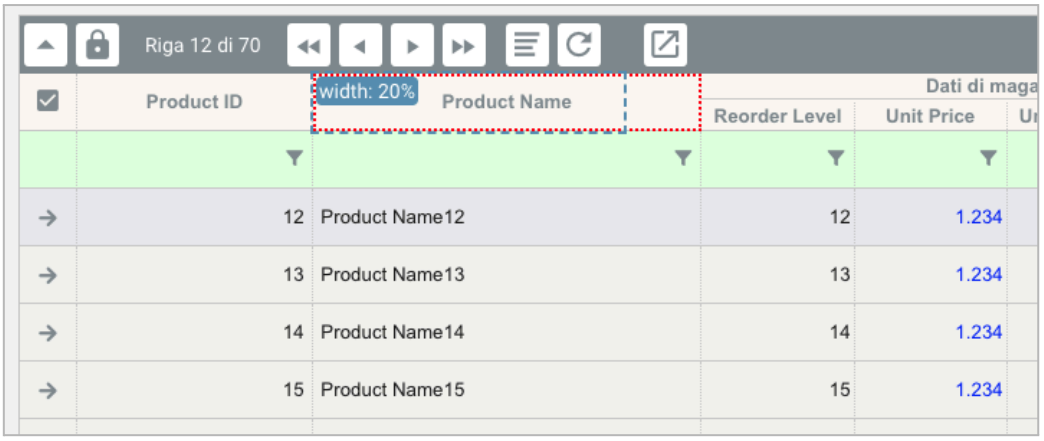
Come per ogni altra datamap è possibile utilizzare anche una query di caricamento dati, oppure specificare le proprietà e caricare i dati direttamente in memoria. Tuttavia la soluzione più flessibile è sempre quella di passare attraverso documenti.

Modifica del layout in griglia

La modifica del layout in griglia avviene tramite l'editor delle videate. Cliccando su una colonna essa viene selezionata nell'albero degli oggetti; cliccando su più colonne tenendo premuto il tasto ctrl, verrà effettuata una selezione multipla.

Trascinando con il drag&drop le intestazioni delle colonne è possibile cambiarne la sequenza: la colonna trascinata apparirà prima di quella su cui viene rilasciata. Inoltre,

posizionando il mouse sul lato destro di una colonna, sarà possibile ridimensionarla. Occorre tenere conto che il ridimensionamento di solito avviene in percentuale, ma è possibile anche gestire la modalità assoluta (pixel). Per maggiori informazioni, vedi il paragrafo Gestione della larghezza e dell'altezza dei campi e delle colonne.

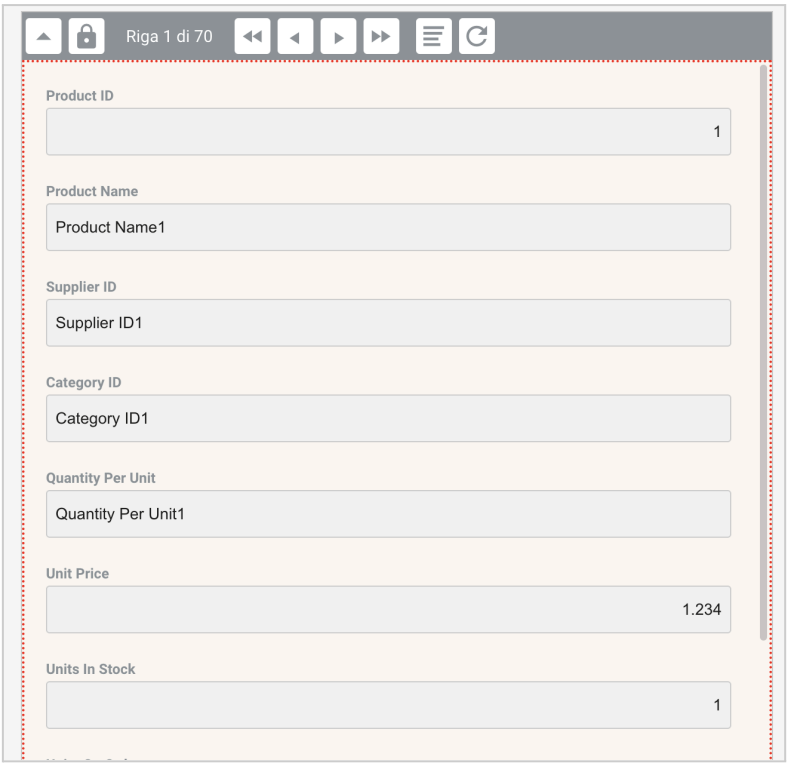


Ridimensionamento di una colonna in pixel

Modifica del layout in dettaglio

Cliccando il pulsante *Cambio Layout* nella toolbar del pannello, il primo dopo i pulsanti di navigazione, si accede alla visualizzazione in dettaglio. Un singolo pannello, infatti, possiede entrambe le visualizzazioni in modo da poter gestire al meglio anche dati molto complessi.

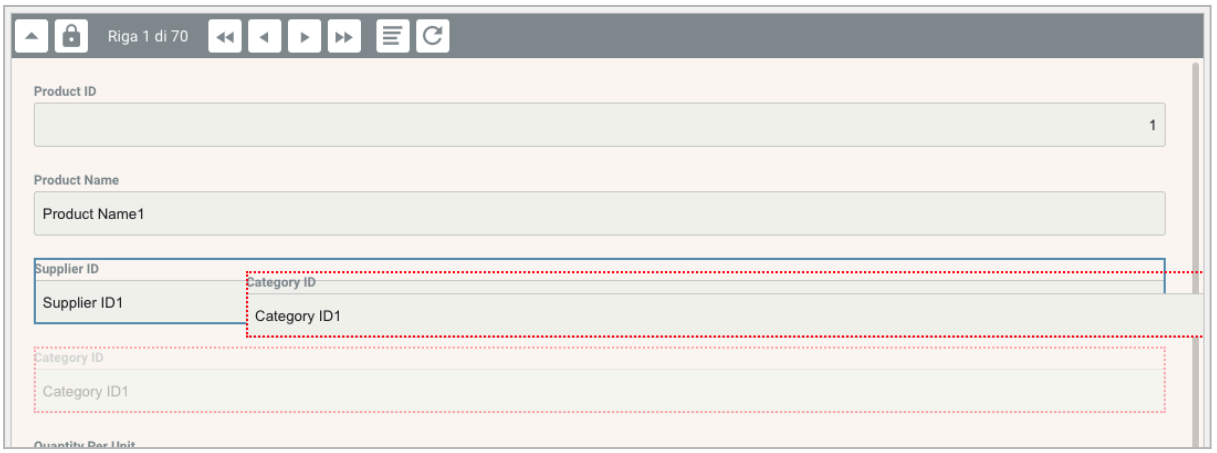
Inizialmente la visualizzazione in dettaglio appare come una lista di campi uno sotto l'altro, come si vede nella figura seguente.



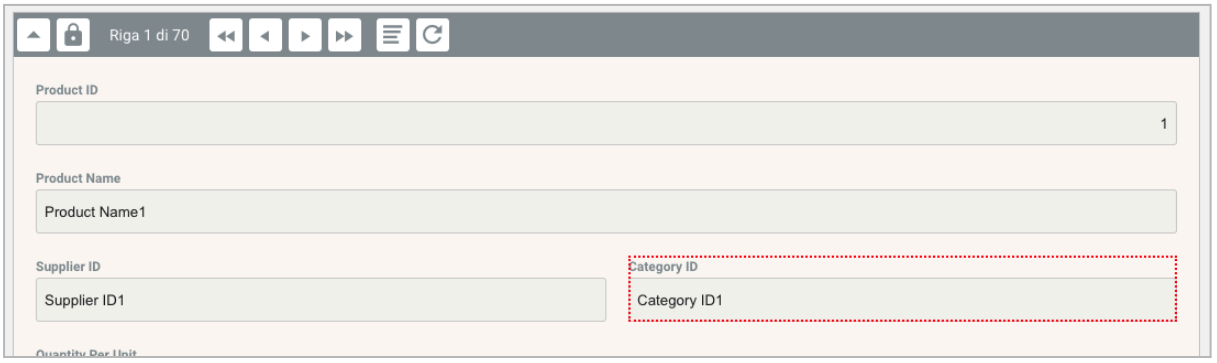
Layout iniziale in dettaglio

Il layout di dettaglio viene gestito come una serie di righe, ognuna delle quali può contenere uno o più campi. All'inizio ogni riga contiene un unico campo. Tramite il trascinamento dei campi con il drag&drop è possibile:

1) Spostare un campo in una riga diversa. I campi presenti nella riga verranno ridimensionati per tenere conto del nuovo campo.

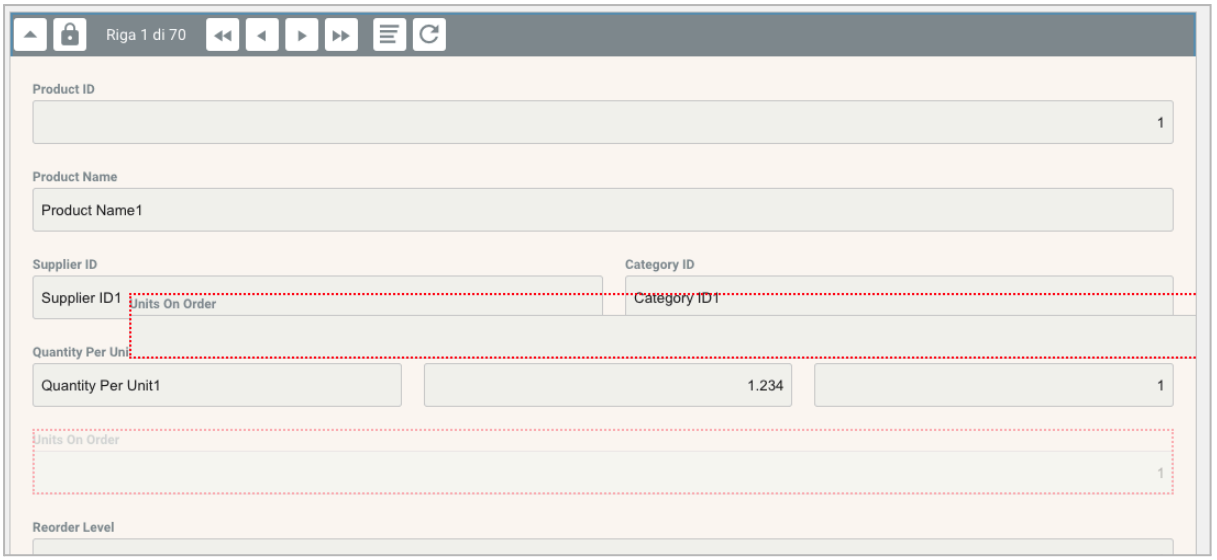


Spostamento del campo Category ID

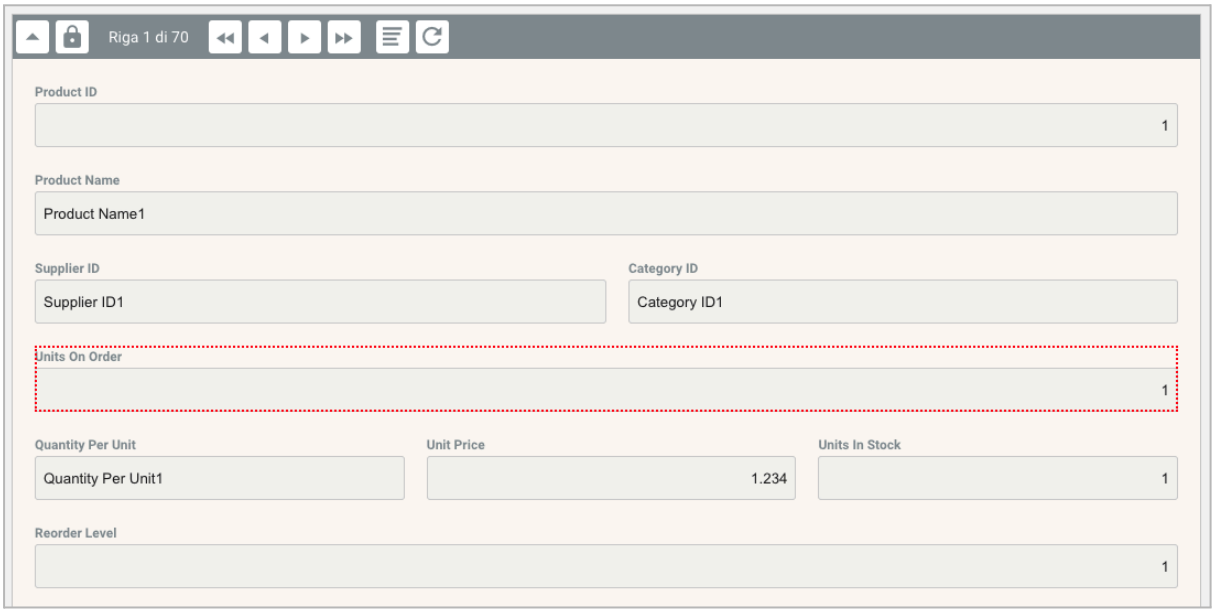


I campi Supplier ID e Category ID condividono la stessa riga.

2) Spostare un campo in mezzo ad altre due righe per creare una nuova riga intermedia.



Spostamento del campo Units On Order in mezzo ad altre righe

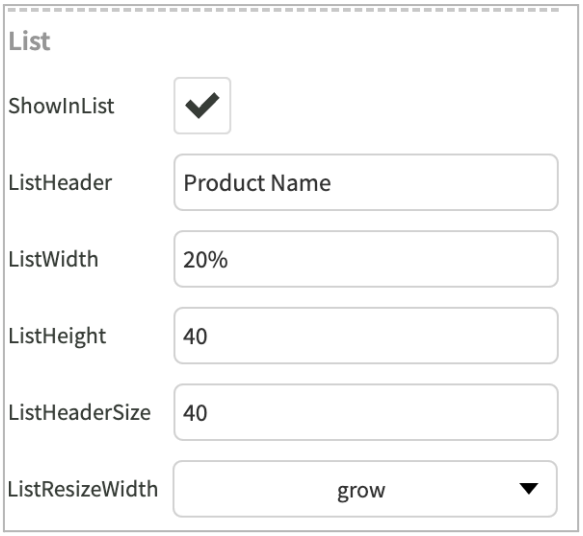


Il campo Units On Order è stato riposizionato

Gestione della larghezza e dell'altezza dei campi e delle colonne

Le dimensioni del pannello e dei campi possono essere specificate in valore assoluto (pixel) o in percentuale, relativamente all'oggetto che li contiene. Questo può avvenire trascinando il bordi del campo oppure impostando le relative proprietà di design time.

Nell'immagine seguente vengono riportate le proprietà della colonna *Product Name*, che nel layout lista ha una larghezza del 20% rispetto al pannello e un'altezza di 40 pixel.



È importante notare la modalità di ridimensionamento (*grow*) che fa sì che la colonna tenda a occupare tutto lo spazio disponibile, fino a riempire l'intera griglia, ma senza mai scendere sotto la dimensione impostata a design time. In presenza di molte colonne, questo comportamento potrebbe far apparire una scrollbar orizzontale nella griglia.

Utilizzando come valore *stretch*, la colonna verrà ridimensionata rispetto al suo valore base cercando di utilizzare l'intera dimensione della griglia senza far apparire la scrollbar orizzontale. In questo caso, se ci sono molte colonne, esse potrebbero risultare molto strette. Se si verifica questa situazione, è sempre possibile impostare un valore per la larghezza minima della colonna tramite lo stile del campo.

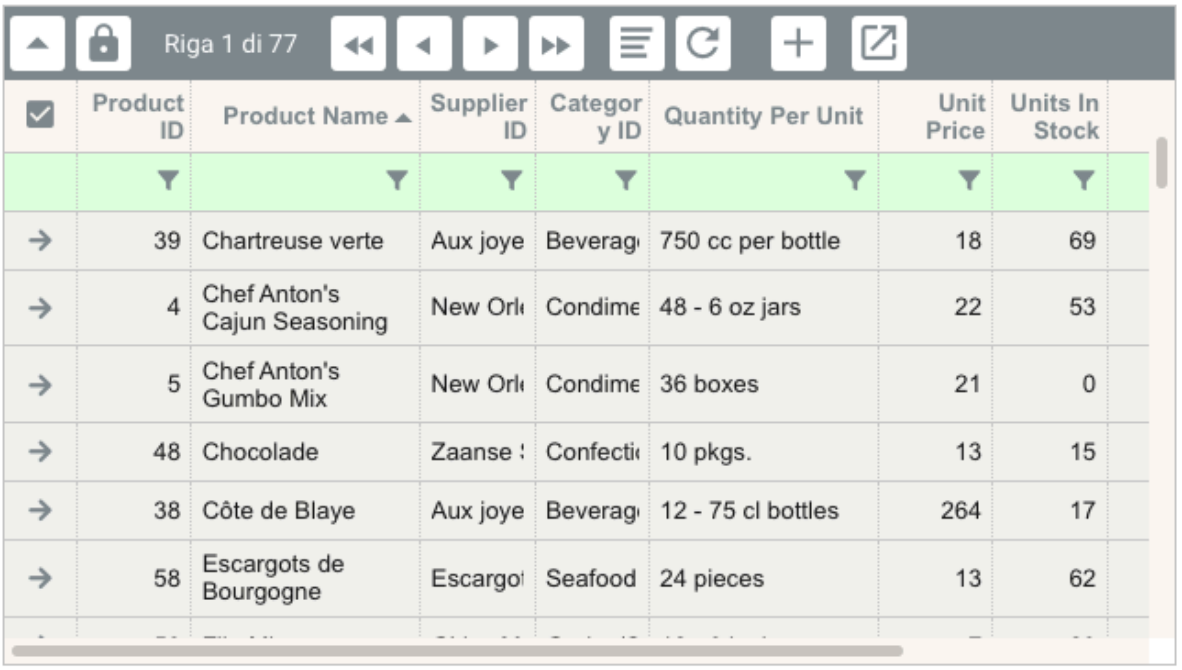
Se infine si utilizza *none* come modalità di ridimensionamento, la colonna manterrà sempre il valore impostato a design time, sia in termini relativi che assoluti.

Per quanto riguarda il dettaglio, dopo aver impostato il layout dei campi tramite il form editor, è possibile aggiustare il posizionamento degli stessi trascinando i bordi del campo oppure tramite le proprietà di design time del campo stesso. In particolare le proprietà *formLeft*, *formTop*, *formRight* e *formBottom* rappresentano i margini del campo rispetto alla riga in cui esso è contenuto, mentre *formWidth* e *formHeight* le rispettive dimensioni. Anche in questo caso le proprietà *formResizeWidth* e *formResizeHeight* stabiliscono le modalità di ridimensionamento.

Campi e colonne multiriga

Normalmente i campi possono contenere testo su una sola riga. Per attivare la possibilità di inserire testi multiriga, è possibile impostare i flag di design time *ListMultiRows* e *FormMultiRows*.

Occorre tenere conto anche che normalmente le righe della lista hanno sempre la medesima altezza definita a design time, quindi è possibile che il contenuto di una cella non sia sempre completamente visibile. È possibile attivare il flag del pannello *RowHeightResize* per fare in modo che l'altezza delle righe sia calcolata in funzione del contenuto.



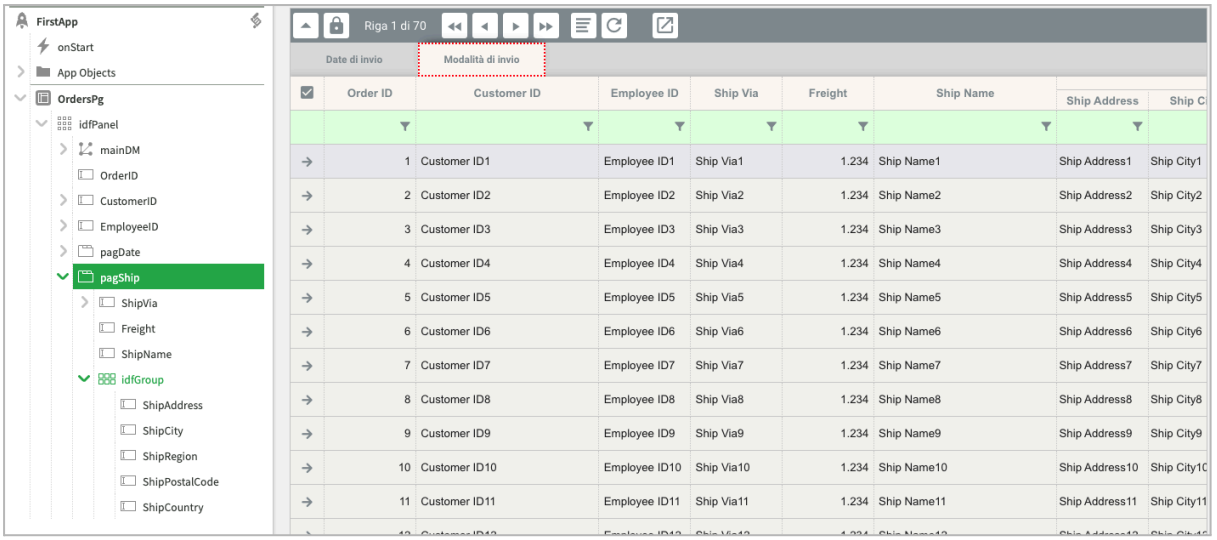
Esempio di pannello con righe ad altezza variabile in funzione del contenuto

Nel layout di dettaglio, i campi mantengono la stessa altezza o si estendono in funzione della politica di ridimensionamento. Selezionando *grow*, il campo si ridimensionerà in altezza in funzione del contenuto; in caso contrario potrebbe apparire la scrollbar.

Raggruppamenti e pagine

I campi dei pannelli possono essere raggruppati tramite la definizione di pagine di campi e di gruppi di campi. Un pannello può quindi contenere elementi di tipo *IdfPanelPage* e *IdfGroup*; è possibile aggiungere tali elementi tramite la barra degli elementi visuali.

Nell'immagine seguente possiamo vedere che alcuni campi sono direttamente contenuti nel pannello (*OrderID*, *CustomerID* e *EmployeeID),* mentre gli altri sono suddivisi nelle pagine *pagDate* e *pagShip*. Inoltre all'interno della pagina *pagShip* alcuni campi sono contenuti all'interno di un elemento *idfGroup*.



Esempio di pannello suddiviso in pagine e gruppi

In questa configurazione, il pannello presenta la toolbar di selezione delle pagine nella parte superiore, subito sotto la toolbar con i comandi del pannello. Ogni pagina contiene i campi comuni, cioè quelli contenuti direttamente nel pannello, e i propri campi.

L'assegnazione dei campi alle varie pagine o gruppi avviene trascinando i campi dall'albero del progetto. Se si visualizza l'anteprima del pannello nel layout di dettaglio, è possibile trascinare i campi direttamente dentro o fuori da un gruppo per cambiarne l'appartenenza.

Gestione delle intestazioni

Le intestazioni dei vari componenti del pannello possono essere modificate tramite le corrispondenti proprietà. In particolare:

Pannelli

Le proprietà *caption* e *icon* del pannello permettono di specificare un titolo e un'icona che compariranno nella toolbar superiore.

Pagine

Le proprietà *caption* e *image* delle pagine di pannello permettono di specificare un titolo e un'icona che compariranno nella zona di selezione delle pagine.

Gruppi

Le proprietà *caption* e *image* dei gruppi di campi permettono di specificare un titolo e un'icona che compariranno nella zona di selezione delle pagine.

Per quanto riguarda il layout di dettaglio, la proprietà *formHeaderPosition* permette di scegliere dove far apparire il titolo del gruppo; mentre la proprietà *formHeaderHeight* permette di specificare l'altezza del titolo in pixel.

Campi

Le proprietà *ListHeader* e *FormHeader* permettono di specificare il titolo dei campi sia in formato lista che in dettaglio. È inoltre possibile scegliere se visualizzare o meno l'intestazione in lista o in dettaglio tramite i flag *ShowListHeader* e *ShowFormHeader.*

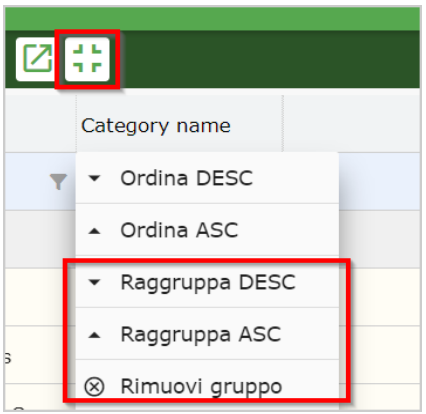
Per quanto riguarda il layout di dettaglio, il flag *formHeaderAbove* permette di scegliere se far apparire il titolo del campo sopra o a sinistra, mentre la proprietà *formHeaderSize* permette di specificare l'altezza del titolo in pixel.

Righe raggruppate

Le righe di un pannello visualizzato nel formato lista possono essere raggruppate in base al contenuto delle sue colonne.
Per attivare i raggruppamenti è necessario impostare a *true* il flag *canGroup* del pannello. Attivando questo flag, nella toolbar del pannello viene reso disponibile il pulsante che permette di attivare il raggruppamento delle righe.

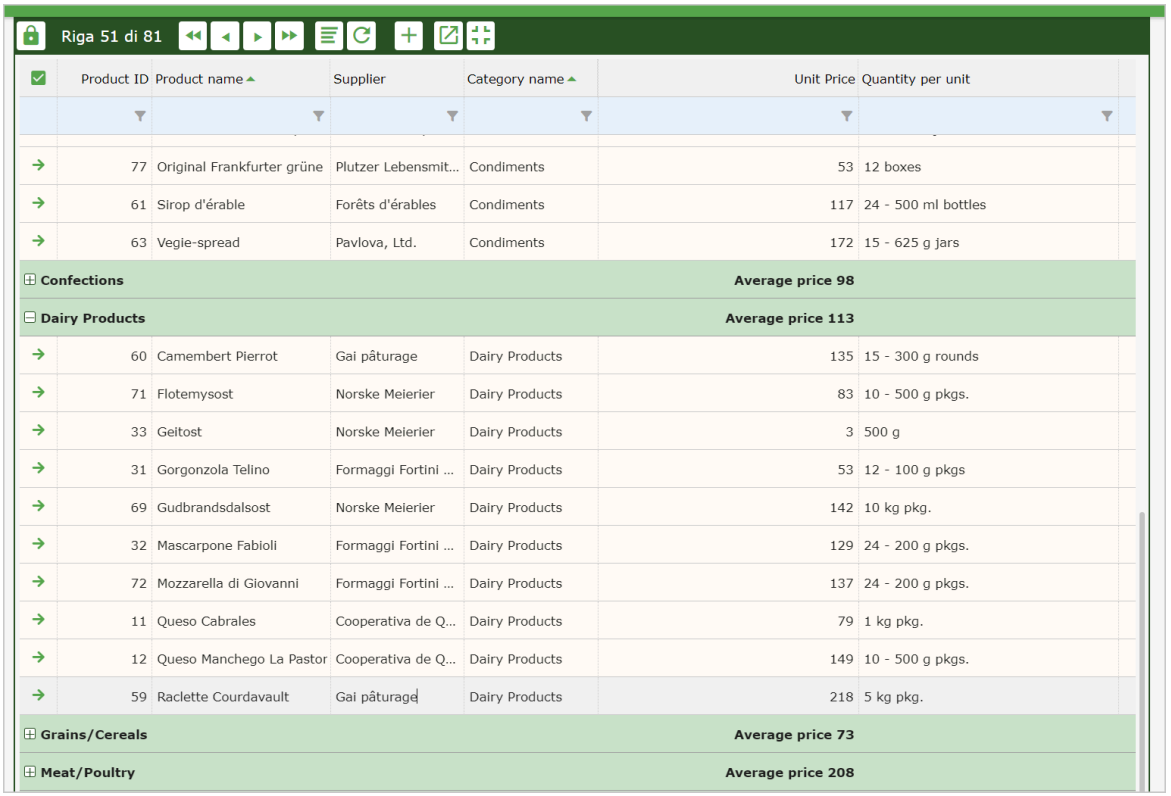
A questo punto l'utente può cliccare sull'intestazione di una o più colonne per selezionare la modalità di raggruppamento che desidera.

È possibile raggruppare ordinando in modalità crescente o decrescente oppure rimuovere un raggruppamento nel caso sia attivo.
Nell'immagine seguente sono evidenziati il pulsante della toolbar utilizzato per disattivare il raggruppamento delle righe e il menu popup che mostra le opzioni di raggruppamento.



Non è possibile raggruppare per la chiave primaria e nemmeno per i campi chiave che mostrano la decodifica di una tabella correlata mediante una combo. Per attivare il raggruppamento in un campo di questo tipo occorre aggiungere un campo unbound derivato dalla tabella correlata.

Per mostrare le righe raggruppate già all'apertura del pannello, è sufficiente specificare la proprietà *groupBy* per la datamap associata al pannello ed impostare a *true* il flag *showGroup* del pannello. Si ricorda che la funzione di raggruppamento non è compatibile con la funzione *Data Paging*.



Product ID	Product name	Supplier	Category name	Unit Price	Quantity per unit
77	Original Frankfurter grüne	Plutzer Lebensmit...	Condiments	53	12 boxes
61	Sirop d'érable	Forêts d'érables	Condiments	117	24 - 500 ml bottles
63	Vegie-spread	Pavlova, Ltd.	Condiments	172	15 - 625 g jars
Confections				**Average price 98**	
Dairy Products				**Average price 113**	
60	Camembert Pierrot	Gai pâturage	Dairy Products	135	15 - 300 g rounds
71	Flotemysost	Norske Meierier	Dairy Products	83	10 - 500 g pkgs.
33	Geitost	Norske Meierier	Dairy Products	3	500 g
31	Gorgonzola Telino	Formaggi Fortini ...	Dairy Products	53	12 - 100 g pkgs
69	Gudbrandsdalsost	Norske Meierier	Dairy Products	142	10 kg pkg.
32	Mascarpone Fabioli	Formaggi Fortini ...	Dairy Products	129	24 - 200 g pkgs.
72	Mozzarella di Giovanni	Formaggi Fortini ...	Dairy Products	137	24 - 200 g pkgs.
11	Queso Cabrales	Cooperativa de Q...	Dairy Products	79	1 kg pkg.
12	Queso Manchego La Pastor	Cooperativa de Q...	Dairy Products	149	10 - 500 g pkgs.
59	Raclette Courdavault	Gai pâturage	Dairy Products	218	5 kg pkg.
Grains/Cereals				**Average price 73**	
Meat/Poultry				**Average price 208**	

I raggruppamenti permettono anche di visualizzare il valore aggregato di uno o più campi. Le nuove proprietà *aggregationFunction* e *aggregationLabel* dei campi di pannello permettono di impostare la funzione di aggregazione e l'etichetta da mostrare prima del valore aggregato.

Le funzioni di aggregazione possibili sono le seguenti:

- *summary* - calcola la somma dei valori della colonna.
- *minimum* - recupera il valore minimo della colonna.
- *maximum* - recupera il valore massimo della colonna.
- *average* - calcola la media dei valori della colonna.
- *median* - calcola la mediana tra il valore massimo e minimo della colonna.
- *range* - calcola la differenza tra il valore massimo e il valore minimo della colonna.
- *variance* - calcola la dispersione dei dati elevata al quadrato rispetto alla media.
- *standardDeviation* - calcola la dispersione dei dati espressa nelle stesse unità dei dati originali.

Proprietà del pannello

Vediamo ora quali ulteriori proprietà sono disponibili nell'elemento pannello. Alcune di queste possono essere impostate solo a design time oppure nel momento in cui l'elemento pannello viene creato. Questo solitamente avviene nell'evento *onLoad* della videata che lo contiene.

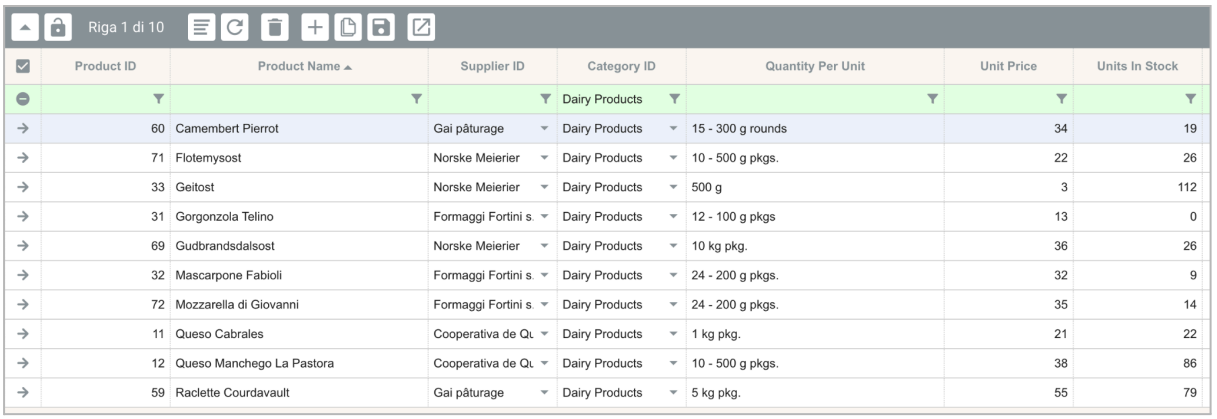
Status

Stato del pannello. Permette di scegliere se il pannello deve effettuare immediatamente la query di ricerca dati appena viene aperto oppure no. Il valore di default è *data*, che significa che il pannello esegue la query; è possibile scegliere *qbe* per aprire il pannello nella modalità di ricerca dati.

Si noti che modificando questa proprietà, anche la corrispondente proprietà *autoload* della datamap principale viene allineata: se lo stato è *data*, *autoload* sarà *true*, altrimenti *false*.

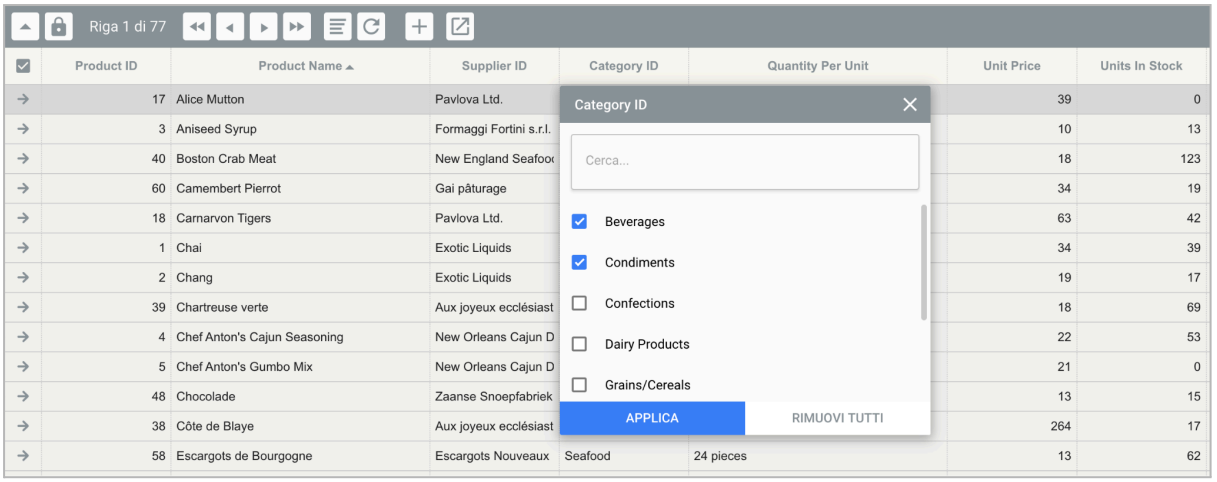
SearchMode

Seleziona una delle tre interfacce di ricerca dati del pannello. Il valore predefinito *row* consiste nell'aggiungere una riga in più in cima alla griglia, dove l'utente può inserire i propri criteri di ricerca. I criteri potranno essere aggiunti direttamente nella cella oppure tramite un popup che guida l'utente nella creazione di criteri complessi.



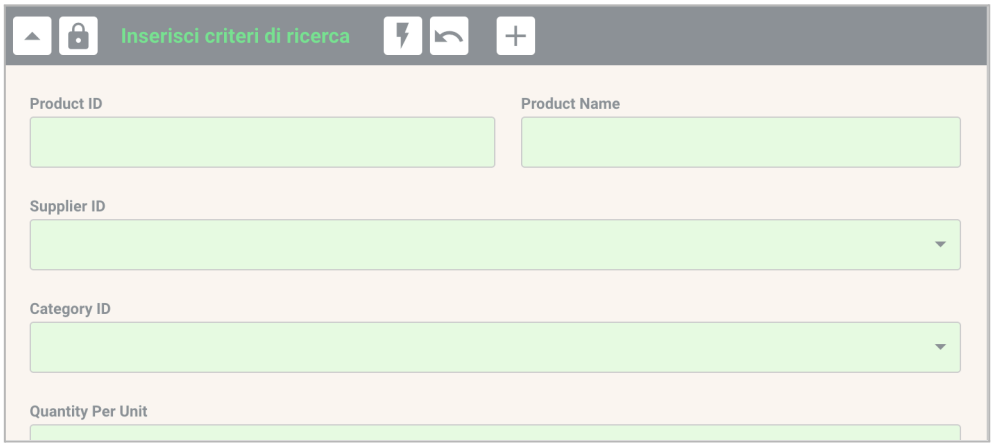
Pannello con searchMode = row

Il valore *header* è simile a *row*, ma non viene inserita la riga aggiuntiva. Quindi l'utente può attivare solo il popup per aggiungere i criteri di ricerca, e questo avviene cliccando sull'intestazione delle colonne della lista. È una modalità simile alle corrispondenti interfacce dei fogli di calcolo, ma meno immediata se il numero di ricerche da fare è elevato.



Pannello con searchMode = header

Infine il valore *toolbar* utilizza una versione alternativa in cui i criteri di ricerca possono essere inseriti sia in lista che in dettaglio (solitamente in dettaglio) scrivendoli direttamente nei campi in cui poi appariranno i dati. È una modalità meno immediata, ma permette di utilizzare tutti i campi del pannello come filtro di ricerca, anche quelli presenti in dettaglio. Selezionando questo valore a design time, la proprietà *status* viene impostata a *qbe*.



Pannello con searchMode = toolbar

Header Height

Rappresenta l'altezza della riga di intestazione della griglia, espressa in pixel.

Active page

Rappresenta la pagina dati attiva del pannello, numerata a partire da zero. Vale solo se il pannello contiene un elemento di tipo *IdfPanelPage*.

Fixed Columns

Rappresenta il numero di colonne fissate della griglia. Se è presente la scrollbar orizzontale e questo numero è maggiore di zero, allora le corrispondenti colonne verranno mantenute fissate a sinistra quando viene effettuato lo scrolling orizzontale della griglia.

Has List

Se questo flag è attivo, allora il pannello presenta il layout griglia, altrimenti no.

Has Form

Se questo flag è attivo, allora il pannello presenta il layout dettaglio, altrimenti no.

Layout

Rappresenta il layout attuale del pannello (lista o dettaglio).

AutomaticLayout

Quando questo flag è impostato a *true*, il pannello gestisce automaticamente la transizione tra lista e dettaglio ed in particolare:

1) Se viene trovata una sola riga, il pannello va automaticamente in dettaglio, altrimenti rimane in lista.
2) Se viene cliccato il pulsante di inserimento, il pannello passa automaticamente in dettaglio e viene sbloccato.
3) Se searchMode = toolbar, la ricerca avviene sempre in dettaglio.
4) Facendo doppio clic su una riga in lista, il pannello passa automaticamente in dettaglio. Il pulsante “lista/dettaglio” della toolbar non appare in nessun layout. Quando il pannello è in modalità dettaglio, appare un pulsante “indietro” nella toolbar.

CanSearch, CanUpdate, CanInsert, CanDelete, CanSort

Attivano o disattivano le corrispondenti funzioni del pannello.

Enable new insert mode

Attivando questo flag, quando si inserisce un nuovo record, la riga entra subito in stato modificato e i valori di default vengono immediatamente esplicitati.

Activate on right click

Se impostato, sarà possibile cliccare sugli attivatori dei campi anche con il tasto destro del mouse.

ShowRowSelector

Mostra o nasconde la colonna laterale per la selezione delle righe.

CanReorderColumn

Attiva la funzionalità di riordino colonne a runtime.

CanResizeColumn

Attiva la funzionalità di ridimensionamento colonne a runtime.

Lockable

Attiva il pulsante per modificare la proprietà *locked* del pannello.

Locked

Rappresenta la modalità di editing attuale del pannello. Se vale *true*, allora è possibile modificare i dati.

RowHeightResize

Se attivo, l’altezza delle righe della griglia dipende dal loro contenuto e non dalle proprietà delle colonne.

ShowMultipleSelection

Se attivo, la colonna iniziale del pannello permette la selezione multipla delle righe.

EnableMultipleSelection

Se attivo, i comandi per mostrare e gestire la selezione multipla saranno attivi per l’utente.

TooltipOnEachRow

Se attivo, i tooltip appariranno anche sulle righe della lista e non solo nell'intestazione.

Caption

Rappresenta il titolo del pannello mostrato nella toolbar superiore.

Icon

Rappresenta l’icona del pannello mostrata a sinistra del titolo.

Collapsed

Rappresenta lo stato di collassamento del pannello. Se vale *true*, apparirà solo la toolbar.

Collapsible

Mostra un pulsante nella toolbar per consentire all’utente di modificare la proprietà *Collapsed* del pannello.

ShowToolbar

Mostra la barra dei pulsanti che l’utente può usare per comandare il pannello.

ShowStatusbar

Mostra un testo sulla destra del titolo che rappresenta lo stato attuale del pannello.

SmallIcons

Utilizza un set di icone più piccole, riducendo la dimensione della toolbar del pannello.

OnlyContent

Nasconde l’intera barra dei comandi superiore. Viene mostrato solo il contenuto del pannello.

Proprietà dei campi di pannello

Enabled

Indica se il campo è abilitato o meno, cioè se l'utente può modificarne il contenuto.

ControlType

Permette di specificare se il campo deve utilizzare un particolare tipo di controllo visuale per la visualizzazione e la modifica dei dati. I possibili valori sono:

- *(vuoto)*: il tipo di controllo viene scelto automaticamente dal pannello. È la scelta consigliata.
- *edit*: verrà sempre utilizzata una edit box.
- *combo*: verrà utilizzata una combo box di tipo autocomplete.
- *check*: verrà utilizzato un check box, utile per campi di tipo boolean.
- *radio*: verrà utilizzata una serie di option button, utile per campi che hanno una lista di valori.
- *button*: verrà utilizzato un pulsante.
- *htmlEditor*: verrà utilizzato un elemento di tipo htmlEditor.
- *blob*: il contenuto del campo verrà mostrato come documento. Se il campo è abilitato, sarà possibile caricare un nuovo documento.

Oltre a questi tipi di controlli è possibile aggiungere e collegare al valore del campo qualunque elemento visuale disponibile. Per maggiori informazioni vedi il paragrafo: Utilizzo di elementi personalizzati nei pannelli.

Placeholder

È un testo descrittivo mostrato quando un campo è vuoto.

Mask

Permette di impostare la maschera di formattazione ed editing del campo. Per le regole di compilazione di questa proprietà fare riferimento alla documentazione della proprietà *mask* dei campi di tipo input.

Badge

Permette di visualizzare un'informazione in formato badge, sul lato destro del campo.

Tooltip

Permette di specificare il tooltip del campo, visualizzato quando il cursore si ferma sull'intestazione.

EnabledInQbe

Se attivo, il campo sarà disponibile come filtro per ricercare i dati.

ValueList

Lista valori associata al campo. Oltre ad una lista valori fissata a design time è possibile definire una query di lookup per recuperare dinamicamente la lista valori per la singola cella.

ActivableDisabled

Permette di specificare se il campo può essere attivato anche quando è disabilitato (read only), oppure solo quando è in modalità di editing.

ActivatorWidth

Permette di specificare la larghezza dell'icona di attivazione del campo. Da usare solo se l'impostazione di default non dà risultati grafici soddisfacenti.

ActivatorImage

Immagine o icona da usare come pulsante di attivazione del campo.

Visible

Visualizza o nasconde la colonna e il campo, sia in lista che nel dettaglio.

ShowInList

Se attivato, include il campo nel layout lista, inserendo la relativa colonna nella griglia.

ListHeader

Titolo della colonna nella lista.

ListWidth

Larghezza della colonna nella lista. Può essere espressa in pixel, ad esempio 100, oppure in percentuale, ad esempio 20%.

ListResizeWidth

Modalità di gestione della larghezza in lista. I possibili valori sono:

- *grow*: se c'è spazio, la colonna verrà allargata per utilizzarlo.
- *stretch*: la colonna può essere allargata oppure ristretta per tentare di eliminare la scrollbar orizzontale.
- *none*: la colonna mantiene sempre la dimensione data.

ListHeight

Altezza delle celle della colonna nella lista espressa in pixel. Le righe della lista saranno alte come la massima *ListHeight* delle colonne. Se però è attivo il flag *RowResizeHeight* del pannello, allora è il contenuto delle celle che viene usato per determinare l'altezza e ogni riga può avere altezza diversa.

ShowListHeader

Indica se l'intestazione della colonna deve essere mostrata o nascosta.

ListMultiRows

Indica se il contenuto delle celle di questa colonna può andare su più righe o meno. Vale per i campi di tipo testuale.

CanHideInList

Indica se l’utente può nascondere questa colonna come parte del sistema di riconfigurazione a runtime dei pannelli.

HiddenInList

Indica se questa colonna è stata nascosta dall’utente tramite il sistema di riconfigurazione a runtime dei pannelli.

CanSort

Indica se questa colonna è ordinabile dall’utente cliccando sull’intestazione. Attenzione: le colonne ID decodificate tramite query di lookup per default non sono ordinabili perché l’ordinamento avverrebbe per ID e non per decodifica.

ShowInForm

Se attivato, include il campo nel layout di dettaglio.

FormHeader

Titolo del campo nel layout di dettaglio.

FormLeft, FormTop, FormRight, FormBottom

Margini del campo, espressi in pixel o in percentuale, riferiti alla riga del dettaglio in cui esso è contenuto e agli altri campi presenti nella stessa riga.

FormWidth, FormHeight

Dimensioni del campo, espressi in pixel o in percentuale. Comprendono sia il campo in sé che la dimensione dell’intestazione.

FormResizeWidth, FormResizeHeight

Modalità di gestione delle dimensioni in dettaglio. I possibili valori sono i medesimi della proprietà *ListResizeWidth* precedente.

FormHeaderSize

Dimensione dell’intestazione espressa in pixel.

ShowFormHeader

Indica se l’intestazione del campo deve essere mostrata o nascosta nel layout di dettaglio.

FormHeaderAbove

Indica se l’intestazione del campo deve essere mostrata sopra o di lato.

FormMultiRows

Indica se il contenuto delle celle di questo campo può andare su più righe o meno. Vale per i campi di tipo testuale.

Alignment

Indica il tipo di allineamento del valore nel campo. Si consiglia di lasciare il valore automatico. Nota: se si desidera impostare allineamenti diversi per lista e dettaglio è possibile impostare il valore *unset*, poi usare l'editor degli stili di lista e di dettaglio.

SuperActive

Se questo flag viene attivato, ogni volta che l'utente clicca un pulsante nel campo o comunque effettua una modifica del contenuto, il valore viene comunicato al server. Solitamente il cambio di valore viene comunicato al momento dell'uscita dal campo.

Multiupload

Se questo flag viene attivato, il campo ammette il caricamento di file.

MaxUploadSize

Dimensione massima dei file che è possibile caricare. Il valore di default è 10 MB.

MaxUploadFiles

Numero massimo di file che è possibile caricare; default 1.

UploadExtensions

Lista delle estensioni dei file che è possibile caricare, separate da virgola.

MaxAutoShowSize

Dimensione massima del contenuto del campo che viene automaticamente mostrata nel dettaglio e nella lista. Default 50 KB. Se il contenuto del campo è più grande, verrà mostrato un link che aprirà il contenuto in una nuova pagina.

Proprietà dei gruppi

Caption

Titolo del gruppo di campi.

Collapsed

Rappresenta lo stato di collassamento del gruppo di campi in dettaglio. Se vale *true*, apparirà solo la barra del titolo. Non ha effetto sul layout lista.

Collapsible

Se attivo, il gruppo potrà essere collassato o espanso cliccando sulla barra del titolo del gruppo stesso.

Visible

Se questo flag viene disattivato, tutti i campi del gruppo sono nascosti. Se invece è attivato, essi saranno visibili in funzione delle regole del singolo campo.

Enabled

Se questo flag viene disattivato, tutti i campi del gruppo sono disabilitati. Se invece è attivato, essi saranno abilitati in funzione delle regole del singolo campo.

Tooltip

Messaggio descrittivo che appare quando il mouse viene posizionato sulla barra del titolo del gruppo.

ListHeaderPosition

Posizione del titolo del gruppo nel layout lista. I possibili valori sono:

- *inner* per indicare che il titolo del gruppo deve apparire nell'intestazione dei campi che lo compongono,
- *none* se il titolo non deve apparire.

FormLeft, FormTop, FormRight, FormBottom

Margini del gruppo, espressi in pixel o in percentuale, riferiti alla riga del dettaglio in cui esso è contenuto e agli altri campi presenti nella stessa riga.

FormHeaderPosition

Posizione del titolo del gruppo nel layout di dettaglio. I possibili valori sono:

- *inner* per indicare che il titolo del gruppo deve apparire all'interno del riquadro del gruppo subito sotto il lato superiore,
- *onBorder*, per indicare che il titolo del gruppo deve apparire sovrapposto al lato superiore del riquadro del gruppo,
- *outer*: per indicare che il titolo del gruppo deve apparire sopra al lato superiore del riquadro del gruppo,
- *none* se il titolo non deve apparire.

FormHeaderHeight

Altezza del titolo del gruppo nel layout di dettaglio espressa in pixel.

Proprietà delle pagine del pannello

Caption

Titolo della pagina.

Image

Icona che appare vicino al titolo della pagina.

Badge

Testo visualizzato come badge sul lato destro del titolo della pagina.

Tooltip

Messaggio descrittivo che appare quando il mouse viene posizionato sul titolo della pagina.

Visible

Stato di visibilità della pagina e dei suoi campi.

Enabled

Se questo flag viene disattivato, tutti i campi della pagina saranno disabilitati. Se invece è attivato, essi saranno abilitati in funzione delle regole del singolo campo.

Personalizzazione grafica del pannello

Sono previsti due livelli per la configurazione grafica dei pannelli. Il primo riguarda un singolo pannello e i suoi oggetti interni, il secondo riguarda tutti i pannelli dell'applicazione.

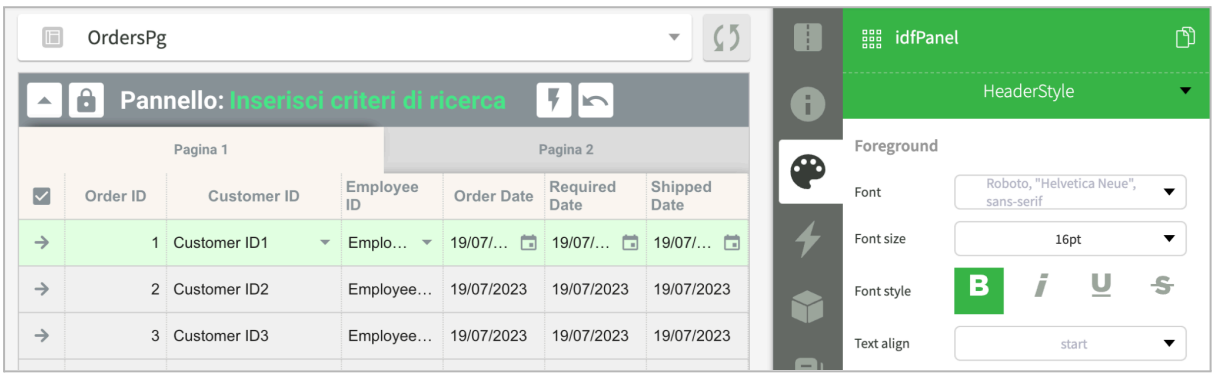
Personalizzazione del singolo pannello

Per modificare la visualizzazione grafica di un singolo pannello, è possibile utilizzare gli stessi strumenti di ogni altro elemento visuale, cioè la definizione di stili in linea e l'applicazione di classi CSS utilizzando la barra degli stili.
Ogni elemento del pannello definisce più stili, in funzione dei suoi componenti interni a cui è possibile applicarli. In particolare:

- l'elemento *idfPanel* definisce *headerStyle* per lo stile della barra del titolo e *ContentStyle* per il contenuto del pannello vero e proprio.
- l'elemento *idfField* definisce *listHeaderStyle* e *formHeaderStyle* per gli stili dell'intestazione nei due layout, *listStyle* per lo stile della colonna nella liste e *formStyle* per lo stile del campo in dettaglio.
- l'elemento *idfGroup* definisce list*HeaderStyle* e *formHeaderStyle* per gli stili dell'intestazione del gruppo nei due layout e *formStyle* per lo stile dello sfondo del gruppo in dettaglio.
- l'elemento *idfPage* definisce solo lo stile di default, che sarà applicato alla barra del titolo della pagina.

Nell'immagine seguente vediamo che è stata cambiata la dimensione del font per l'intestazione del pannello usando la barra degli stili e selezionando *HeaderStyle*.

Si ricorda che oltre all'applicazione di stili in linea è possibile associare agli elementi classi CSS definite nei propri fogli di stile.



Personalizzazione di tutti i pannelli dell’applicazione

La modifica dello stile grafico di tutti i pannelli dell’applicazione avviene modificando il foglio di stile base dei pannelli contenuto nella risorsa *panel CSS*, che si trova nella libreria *IDFWidgets*, classe *IdfPanel*. Se nella sezione librerie del proprio albero del progetto non appare la libreria *IDFWidgets*, occorre cliccare il pulsante *Mostra/Nascondi librerie importate* posizionato sulla sinistra del pulsante *Aggiungi* nella barra del titolo dell’albero degli oggetti.

Il CSS di base dei pannelli deve essere personalizzato aggiungendo le variazioni al proprio foglio di stile, senza modificare l’originale.

Il CSS è diviso in due zone. Nella prima sono contenute le variabili che definiscono i colori degli elementi del pannello. Modificando tali colori si otterrà immediatamente l’allineamento grafico al proprio set di colori.

```
/*****   COLOR PERSONALIZATION *****/

:root {
  /* Toolbars, buttons, tooltip */
  --col-back-primary: #8a9399;
  --col-fore-primary: #ffffff;

  /* Toolbar buttons, popup buttons */
  --col-back-secondary: #ffffff;
  --col-fore-secondary: #8a9399;
```

Segmento di CSS di base dei pannelli che definisce i colori

Se si desidera modificare altri aspetti della grafica dei pannelli oltre ai colori è possibile modificare l’intero CSS di base. Si consiglia di utilizzare gli strumenti di ispezione del browser per vedere quali classi sono utilizzate nelle varie parti del pannello.

Personalizzazione delle stringhe utilizzate nel pannello

Tutte le stringhe che compaiono all'interno di un pannello sono definite nella classe *Client.IdfResources*. Le lingue definite nella libreria *IdfWidgets* sono italiano e inglese. È possibile modificare il testo di una o più stringhe delle lingue di default oppure gestire nuove lingue.

Per ogni lingua da gestire deve essere definito un oggetto *Client.IdfResources.-msg_[langCode]*, dove *[langCode]* è il codice di due lettere che identifica la lingua. Tale oggetto contiene tutte le stringhe tradotte nella lingua indicata dal nome dell’oggetto.

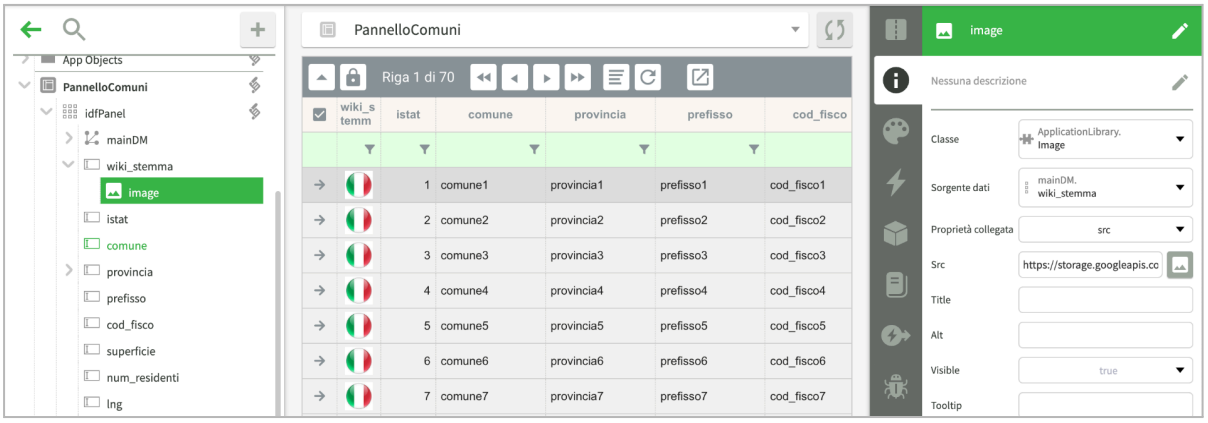
Per vedere come costruire questo oggetto occorre fare riferimento alla risorsa *panelStringsTemplate* contenuta nella classe *IdfPanel* della libreria *IDFWidgets*. Per definire una nuova lingua è sufficiente copiare il testo della risorsa in un proprio metodo da chiamare nell’evento *onStart* dell’applicazione.

Utilizzo di elementi personalizzati nei pannelli

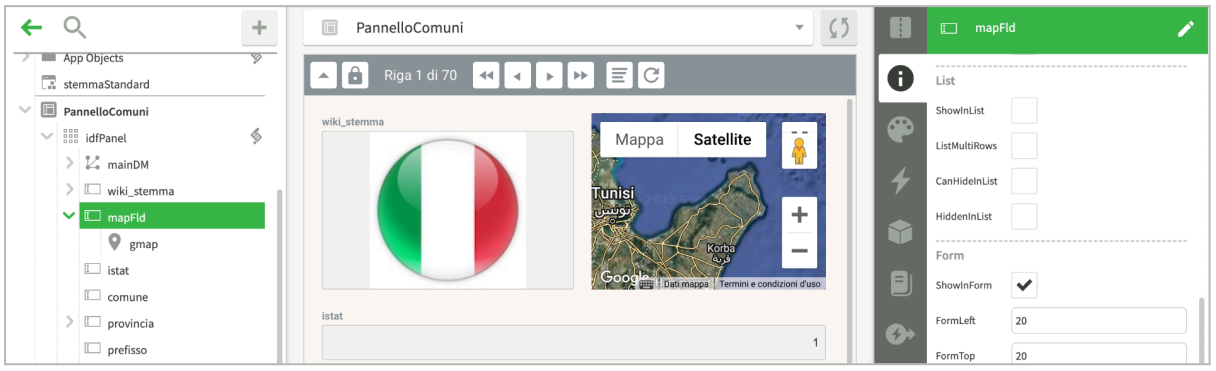
Una delle caratteristiche più interessanti dei pannelli è la possibilità di utilizzare elementi visuali di ogni tipo all'interno dei campi del pannello stesso.

Il caso d'uso più comune è relativo alla visualizzazione del dato contenuto in una colonna o in un campo tramite un elemento diverso da quelli standard. Se, ad esempio, una proprietà di un documento contiene l'indirizzo di un'immagine, si potrebbe visualizzare questo dato in un elemento immagine piuttosto che in un controllo di tipo testo.

Nell'immagine seguente vediamo un esempio: nel campo di pannello *wiki_stemma* è stato aggiunto un elemento *image*, poi nel pannello delle proprietà questo elemento è stato collegato alla sorgente dati *wiki_stemma*, in modo che la proprietà *src* dell'immagine venga impostata al valore della proprietà corrispondente del documento.



Un secondo caso d'uso per gli elementi personalizzati è la visualizzazione di dati aggiuntivi del documento utilizzando nuovi campi visibili solo nel layout di dettaglio e non collegati alle proprietà della datamap. Vediamo di un esempio di questo caso in cui viene usata una mappa per visualizzare la posizione del comune selezionato dalla griglia.



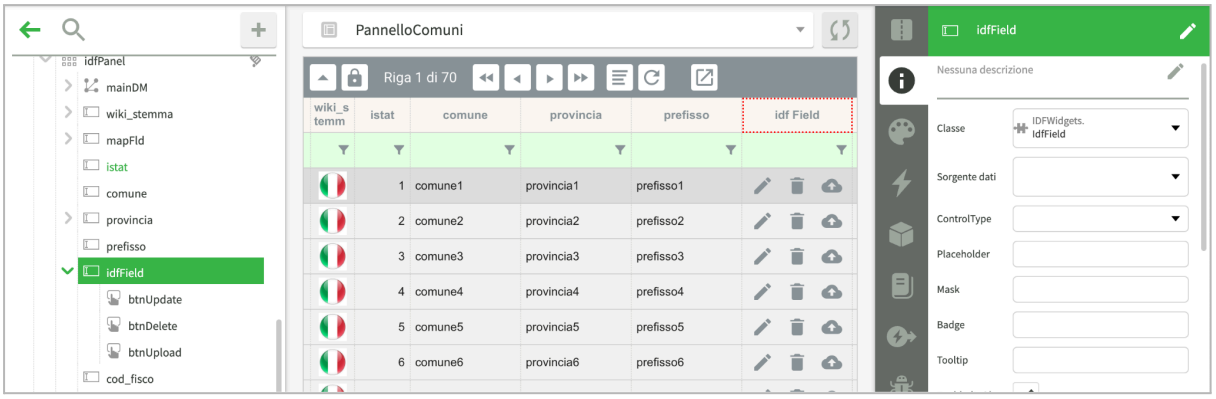
Si noti nell'immagine che il campo *mapfld* ha il flag *ShowInList* disabilitato, quindi la mappa compare solamente nel layout di dettaglio.

Siccome il campo *mapFld* non è collegato alle proprietà della datamap principale del pannello, il suo contenuto non cambia al variare dei dati presenti nel layout di dettaglio. È

quindi necessario implementare l'evento del pannello *onRowChanged* per allineare le proprietà della mappa. Il codice dell'evento potrebbe essere il seguente:

```
$idfPanel.onRowChanged = function(event)
{
  $gmap.center = {lat : event.newRow.lat, lng : event.newRow.lng};
};
```

Un terzo caso d'uso per gli elementi personalizzati è l'inserimento di controlli particolari per operare sulla riga attiva della griglia. In questo caso è possibile aggiungere un nuovo *idfField* al pannello, non collegato alle proprietà della datamap principale. All'interno di questo campo è poi possibile aggiungere pulsanti o altri controlli come mostrato nell'immagine seguente:



Nell'evento *onClick* dei pulsanti è possibile fare riferimento alla riga del pannello usando la proprietà *row* della datamap principale. Il pulsante di cancellazione, ad esempio, può essere gestito tramite il seguente codice:

```
$btnDelete.onClick = function(event)
{
  $mainDM.row.deleted = true;
};
```

I pannelli a runtime

In questo paragrafo vediamo le dinamiche del pannello, cioè come controllare il suo comportamento a runtime tramite eventi e metodi. Occorre tenere conto, tuttavia, che spesso si farà riferimento alla datamap principale del pannello che ne coordina l'accesso ai dati.

Un esempio di utilizzo degli pannelli è nel progetto di esempio North Wind Web che è possibile trovare nella sezione *Documentazione* della Console di Instant Developer Cloud.

Fase di ricerca e visualizzazione dei dati

Quando un pannello viene aperto, è possibile gestire automaticamente o manualmente il caricamento iniziale dei dati. A tal fine è possibile impostare a design time la proprietà *status* del pannello a *qbe* invece che *data* per evitare il caricamento iniziale.

Il valore di design time della proprietà *status* viene usato per allineare la proprietà *autoload* della datamap principale. Nel caso in cui i dati non siano stati caricati, è possibile farlo chiamando il metodo *load* sulla datamap principale. In questo modo, se si vogliono impostare filtri di default sulla datamap è possibile usare il metodo *addFilter* prima di chiamare *load*.

Occorre tenere presente che il pannello manipola i filtri della datamap in funzione dei filtri inseriti dall'utente. Se quindi l'utente cancella tutti i filtri, verranno eliminati anche quelli. Se pertanto si desidera aggiungere filtri permanenti, si consideri di impostarli nell'evento *beforeLoad* della datamap principale.

In funzione della proprietà *searchMode*, i campi di ricerca saranno visibili o meno. Se sono sempre visibili, il pannello potrà eseguire ricerche multiple senza cambiare stato. In ogni caso, è possibile investigare il valore inserito dall'utente per la ricerca tramite il metodo *getFilter* della datamap.

Configurazione della visualizzazione delle righe

In molti casi è necessario modificare lo stile o altre proprietà della singola riga della cella in funzione del contenuto della stessa. A tal fine è possibile utilizzare l'evento *onRowComposition* del pannello. Esso viene notificato quando è necessario aggiornare la visualizzazione delle righe del pannello e, come l'analogo evento delle datamap, permette di configurare lo stato degli elementi della riga per cui viene chiamato.

Se, ad esempio, volessimo evidenziare in rosso i nomi dei prodotti sotto scorta, potremmo utilizzare il seguente codice:

```
$idfPanel.onRowComposition = function(event)
{
  event.template.ProductName.style.color =
          (event.row.UnitsInStock < event.row.ReorderLevel) ? "red" : "";
};
```

Se si volesse evidenziare in rosso tutti i campi della riga si potrebbe usare il seguente codice:

```
let c = (event.row.UnitsInStock < event.row.ReorderLevel) ? "red" : "";
for (let fn in event.template) {
  let f = event.template[fn];
  f.style.color = c;
}
```

Le proprietà configurabili degli elementi del template passato tramite il parametro *event* sono le seguenti:

- *text*: quello che viene visualizzato nella cella.
- *enabled*: indica se la cella è abilitata alla modifica o meno (in caso di pannello sbloccato).
- *visible*: indica se la cella è visibile o meno.
- *style*: stile CSS della cella e del suo contenuto.
- *alignment*: allineamento della cella. Usare i valori della lista *App.IdfField.alignments*.
- *badge*: testo che appare come badge della cella.
- *mask*: maschera di inserimento e visualizzazione della cella.
- *errorText*: testo di errore della cella.

Accesso ai dati della riga attiva

Se si desidera accedere ai dati della riga attiva del pannello è possibile farlo in diverse modalità a seconda del momento in cui si vuole fare questa operazione.

Se si vuole accedere ai dati della riga attiva mentre essa cambia occorre utilizzare l'evento *onRowChanged* che viene notificato sia al cambio riga, sia se i dati della riga attiva cambiano, ad esempio perché l'utente li sta modificando. Nei parametri passati dall'evento abbiamo sia la nuova riga che la precedente. Il seguente codice modifica il titolo del pannello in funzione del nome del prodotto presente sulla riga attiva.

```
$idfPanel.onRowChanged = function(event)
{
  this.caption = event.newRow.ProductName;
};
```

Si ricorda che se il pannello è basato su documenti, è possibile accedere al documento presente sulla riga attiva tramite la proprietà *document* della riga; nel caso precedente sarebbe stato *event.newRow.document*.
Se si vuole accedere ai dati della riga attiva in modo statico, cioè fuori da contesti in cui la riga attiva cambia, è possibile accedere direttamente alla riga attiva della datamap principale, ad esempio con il codice seguente:

```
$button.onClick = function(event)
{
  app.alert($mainDM.row.ProductName);
};
```

Si tenga presente che la datamap potrebbe non avere una riga attiva; in tal caso *$mainDM.row* vale *null*.

Infine se si accede alla riga attiva da elementi interni al pannello, è possibile usare anche la proprietà *this.row*, che viene impostata sugli elementi della riga del pannello stesso. Per leggere i dati al momento dell'attivazione di un campo di pannello, è quindi possibile usare il seguente codice:

```
$QuantityPerUnit.onActivated = function(event)
{
  app.alert(this.row.ProductName);
};
```

Selezione multipla

I pannelli supportano anche la selezione multipla dei dati. Tale opzione è attivabile sia tramite i comandi di interfaccia utente che da codice.

Per consentire all'utente di attivare la selezione multipla, è necessario impostare a *true* la proprietà *enableMultipleSelection* del pannello. Per attivarla da codice è necessario impostare a *true* la proprietà *showMultipleSelection*.

Le righe attualmente selezionate nel pannello sono controllabili tramite interfaccia utente o da codice. In questo secondo caso, è possibile usare i metodi di pannello *changeSelection* per cambiare la selezione di tutte le righe, oppure *setRowSelected* per modificare lo stato di selezione di una singola riga.

Se, ad esempio, si desidera attivare o disattivare la selezione di una riga cliccando su un campo della stessa, è possibile utilizzare il seguente codice:

```
$ProductName.onActivated = function(event)
{
  let sel = $idfPanel.isRowSelected($mainDM.row);
  yield $idfPanel.setRowSelected($mainDM.row, !sel);
};
```

Per permettere di personalizzare la modifica della selezione delle righe, il pannello notifica l'evento *onChangeSelection* tutte le volte che una riga cambia stato di selezione oppure se è stato usato un comando di cambiamento complessivo.

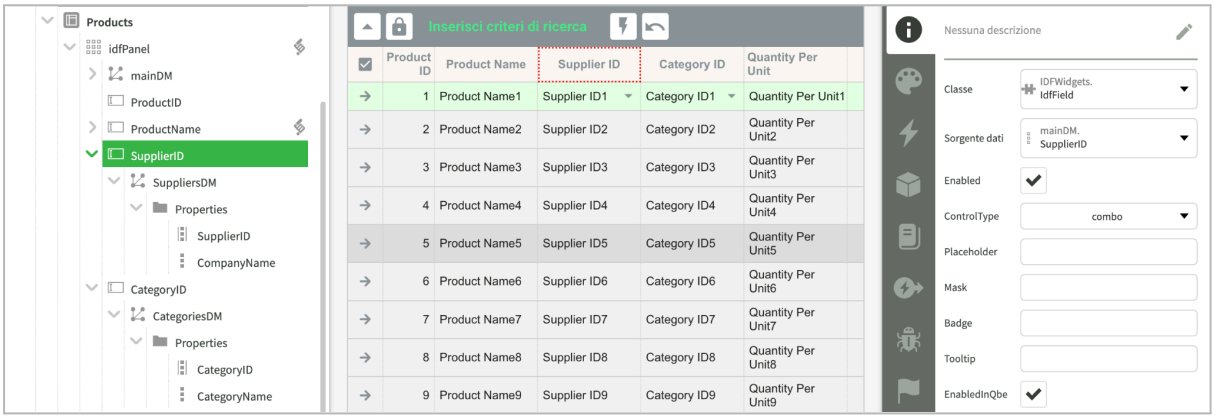
Si noti infine che alcuni comandi di pannello, come la cancellazione o l'esportazione, agiscono sulle righe selezionate se la selezione multipla è mostrata a video.

Query di decodifica e lookup

Molto spesso i pannelli devono mostrare dati di tabelle che contengono referenze ad altre tabelle. Ad esempio, in una tabella degli ordini sarà presente il campo *ID Cliente*, che rappresenta la referenza al cliente che ha effettuato l'ordine.

In questi casi generalmente non si desidera mostrare il dato così com'è, ma si preferisce effettuare una decodifica e portare a video una o più proprietà dell'oggetto referenziato. Nell'esempio precedente si potrebbe mostrare la denominazione del cliente che ha effettuato l'ordine.

Anche nella fase di modifica dei dati, per selezionare i dati desiderati non è possibile inserire direttamente il valore dell'ID Cliente, ma occorre reperirlo tramite un'operazione di lookup sulla tabella dei clienti.
Per poter organizzare queste operazioni, è possibile aggiungere una datamap all'interno di un campo del pannello. Impostando la proprietà *controlType* del campo al valore *combo*, si otterrà un controllo di tipo autocomplete che utilizza la datamap interna come sorgente di valori.



i campi SupplierID e CategoryID hanno una datamap interna per lookup e decodifica

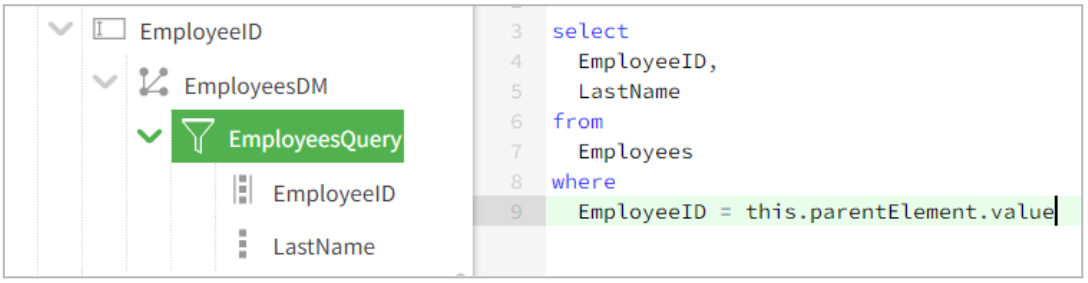
In questa configurazione, il pannello utilizza le datamap per operazioni sia di decodifica che di lookup. In particolare, quando i dati devono essere mostrati a video, la datamap estrae una sola riga per esprimere la decodifica. Quando l'utente vuole modificare il valore del campo tramite lookup, il pannello usa la datamap interna togliendo il vincolo sul campo in fase di editing per estrarre la lista dei possibili valori e permettere all'utente di sceglierli.

Le datamap di lookup possono essere basate sia su query che su documenti. In entrambi i casi devono estrarre due colonne, la prima deve contenere il valore da inserire nel campo mentre la seconda la decodifica. Nell'immagine precedente viene mostrata la configurazione delle proprietà di queste datamap.

Se si desidera modificare le proprietà selezionate in una datamap di lookup creata automaticamente dal pannello, è possibile modificare la proprietà *Derivato da* della datamap indicando quale proprietà del documento deve essere collegata.

Nel caso si voglia utilizzare una concatenazione di campi è possibile utilizzare una proprietà unbound valorizzata nell'evento *afterLoad* del documento e poi utilizzare quella come proprietà selezionata dalla datamap di lookup.

Se la datamap di lookup è basata su query, occorre specificare il legame con il campo di pannello che la contiene referenziandolo tramite *this.parentElement.value,* come mostrato nell'immagine seguente.



Datamap di lookup basata su query

Le datamap di lookup presentano alcune proprietà che ne configurano il funzionamento. In particolare:

- *LookupCacheMode*: questa proprietà permette alla datamap di memorizzare i valori ottenuti in fase di decodifica per poterli riutilizzare risparmiando ulteriori query. I possibili valore sono:
  - *none:* nessuna operazione di cache viene effettuata,
  - *single:* viene memorizzato e riutilizzato il valore restituito della query di decodifica di ogni riga. È nella maggior parte dei casi il valore migliore perché permette di risparmiare query senza sovraccaricare la memoria del pannello.
  - *full*: l'intera lista di possibili valori viene caricata una sola volta all'apertura del pannello e nessuna query viene più effettuata per le operazioni di decodifica. Questo valore è il migliore se il numero di righe della tabella da decodificare è basso, nell'ordine delle decine di righe.
- *EnableValidation:* se questa proprietà è *true*, in fase di modifica è necessario che l'utente selezioni un valore. Non è ammesso il valore *null* per questo campo.

È possibile personalizzare il comportamento delle datamap di lookup e decodifica come per ogni altra datamap, intercettandone gli eventi, in particolare *beforeLoad* e *onSearch*. Infine si ricorda che se si utilizza una datamap di lookup con cache dei dati, è possibile resettare il contenuto della cache usando il metodo *refreshLookup* del campo che contiene la datamap di lookup.

Fase di modifica e salvataggio dei dati

I pannelli ammettono le operazioni di modifica, inserimento e cancellazione sia nella modalità lista che dettaglio. Tali operazioni vengono normalmente pilotate tramite i comandi della toolbar nel seguente modo.

Per default, quando appaiono i dati il pannello è bloccato. Si può modificare questa impostazione disattivando la proprietà *locked* del pannello. Tuttavia in questo caso è possibile che l'utente modifichi i dati senza volerlo.

Le operazioni di modifica e cancellazione avvengono nel layout attuale dopo aver sbloccato il pannello tramite il relativo comando della toolbar. L'inserimento può avvenire anche a

pannello bloccato, se la proprietà del pannello *enableInsertWhenLocked* è attiva. Se *automaticLayout* è *true*, cliccando il pulsante di inserimento il pannello viene portato in dettaglio.

È possibile personalizzare questi passaggi intercettando gli eventi *onLockingChanging* e *onStatusChanged* del pannello.

Se si desidera personalizzare la gestione delle modifiche ai dati, occorre invece utilizzare gli eventi della datamap o del documento, se la datamap è basata su documenti. Si rimanda alla documentazione relativa per maggiori informazioni.

Modifica programmatica dei dati

Durante la fase di modifica dei dati è possibile modificare i dati anche in maniera programmatica, sempre utilizzando l'accesso alla riga attiva del pannello tramite la proprietà *row* della datamap principale.

Vediamo come esempio l'utilizzo di una videata di lookup esterna per recuperare un'informazione necessaria nel pannello attuale. Come esempio di partenza utilizziamo un pannello sulla tabella *RigheOrdine* che necessita di una videata di lookup sulla tabella *Ordini* per scegliere il numero d'ordine.

Attivando il campo *OrderID* del pannello *RigheOrdine* apriamo la videata di lookup, con il seguente codice:

```
$OrderID.onActivated = function(event)
{
  App.Pages.push(app, App.Ordini, {popup : true, modal : true});
};
```

Dopo aver scelto l'ordine giusto, chiudiamo la videata di lookup con il seguente codice:

```
$idfPanel.onDblclick = function(event)
{
  App.Pages.pop(app, 1, {row : $mainDM.row});
};
```

e infine dalla videata *RigheOrdini* applichiamo alla riga corrente l'ID ordine selezionato con la videata di lookup intercettando l'evento *onBack*.

```
App.RigheOrdine.prototype.onBack = function(sender, info)
{
  if (info.row && info.row.OrderID) {
    $mainDM.row.OrderID = info.row.OrderID;
  }
};
```

Caricamento e visualizzazione di documenti (blob)

I pannelli mettono a disposizione la funzionalità di caricamento, visualizzazione e scaricamento di documenti (file) in modo completamente automatico. Sarà sufficiente aggiungere un campo di tipo blob al database e configurare le opzioni del campo di pannello corrispondente. In particolare:

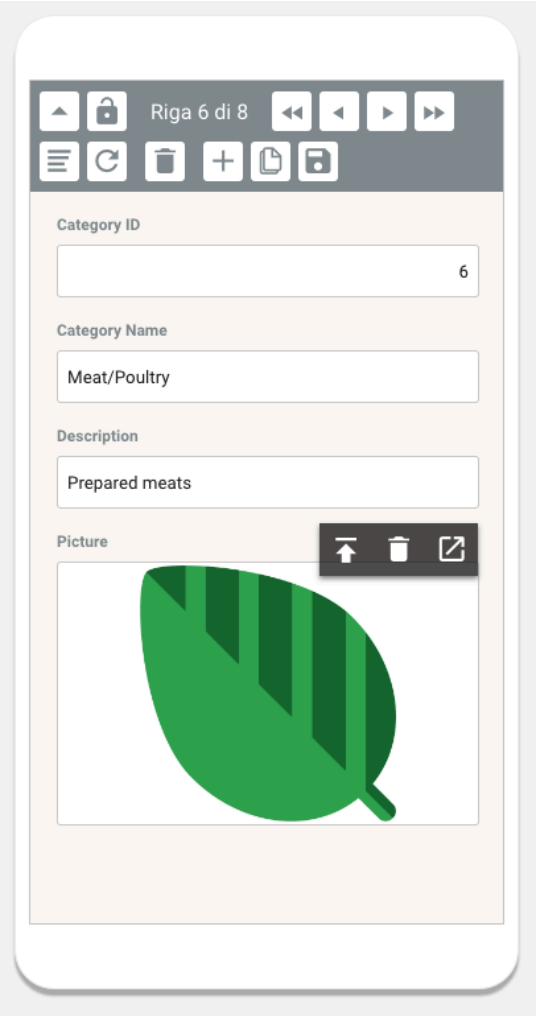
- *maxUploadSize*: dimensione massima caricabile in byte. Default 10 MB.
- *maxAutoShowSize*: dimensione massima visualizzabile al volo nel pannello. Default 50 KB. Se il contenuto del blob è maggiore di questa dimensione, verrà mostrato un link per la visualizzazione in una nuova pagina.

Il blob può essere visualizzato e modificato sia in lista che in dettaglio. Posizionando il cursore su un campo blob apparirà una toolbar che ne permette la gestione. La toolbar contiene i seguenti pulsanti:

- *Carica*: permette di caricare un nuovo file nel campo. Disponibile solo se il campo non è di sola lettura.
- *Cancella*: Svuota il contenuto del campo, eliminando il file. Disponibile solo se il campo non è di sola lettura.
- *Apri in nuova pagina*: mostra il contenuto del campo in una nuova pagina browser.

I file vengono caricati nella datamap in formato *ArrayBuffer* e poi vengono salvati nel database nel campo blob collegato. È quindi possibile interagire tramite l'evento *onUpload* del campo di pannello per manipolare il contenuto del blob in modo diverso, ad esempio salvando il blob sul file system invece che direttamente nel database.

Si segnala infine che i campi blob mostrati in anteprima nel pannello o in una nuova pagina vengono salvati come file temporanei nel file system del server con un nome casuale per poter essere visualizzati dal browser. Al termine della sessione i file temporanei vengono rimossi dal sistema.



Esempio di campo blob in un pannello

Uso di campi statici con funzione multi-upload

I pannelli permettono di realizzare facilmente superfici di caricamento di file non direttamente collegati al database. A tal fine è necessario aggiungere un nuovo campo al pannello non collegato ad alcuna proprietà della datamap e non mostrato in lista, cioè con il flag *ShowInList* non selezionato. A questo punto è possibile attivare il flag *MultiUpload* nelle proprietà del campo ed impostare eventualmente le proprietà *MaxUploadSize*, *MaxUploadFiles* e *UploadExtensions* per limitare numero, dimensione e tipo di file che sarà possibile caricare.

A runtime, il caricamento avviene cliccando sul campo oppure trascinando i file dal computer nel campo. Lato server viene notificato l'evento *onUpload* che contiene i file caricati. Essi dovranno essere gestiti o cancellati dal file system. Se non si esegue nessuna operazione i file rimarranno nel file system del server per sempre.

Si noti che il campo statico gestisce solo il caricamento dei file, quindi non ha alcun cambiamento grafico durante e dopo questa operazione. È tuttavia possibile utilizzare la proprietà *innerHtml* del campo per inserire testi o altri elementi nel campo.

Coordinamento di più pannelli

All'interno dell'interfaccia utente della propria applicazione è possibile coordinare il contenuto di più pannelli. Questo avviene normalmente assegnando la proprietà *document* o *collection* della datamap principale dei pannelli collegati.

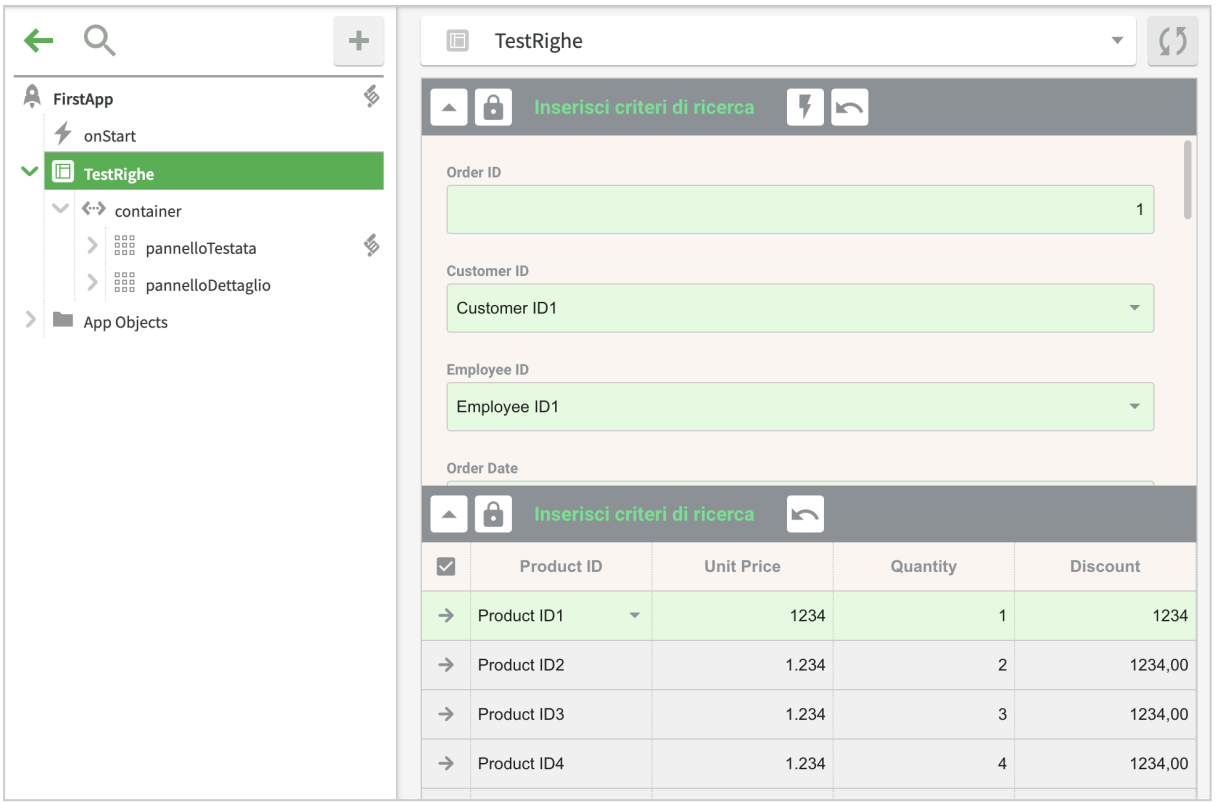
Master detail

Vediamo ora una videata che contiene la lista degli ordini e, per ognuno di essi, l'elenco delle righe corrispondenti.

Il coordinamento fra i due pannelli viene gestito dall'evento *onRowChanged* del pannello di testata, tramite il seguente codice:

```
$pannelloTestata.onRowChanged = function(event)
{
  let o = event.newRow.document; // type:NwindLibreria.Orders
  yield o.OrderDetails.load();
  $pannelloDettaglio.mainDM.collection = o.OrderDetails;
};
```

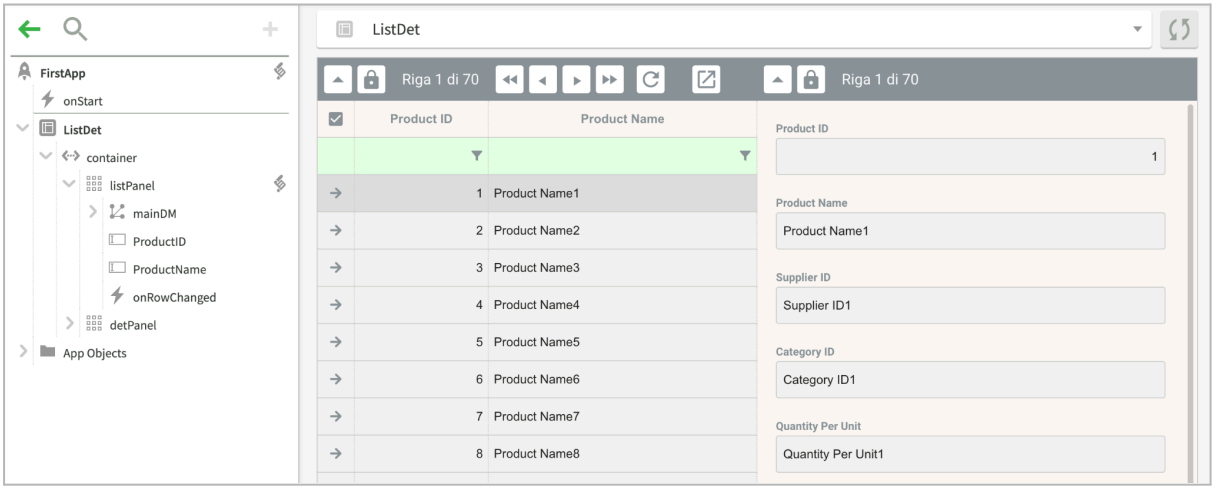
Si noti che in questo caso è necessario caricare la collection delle righe perché quando il pannello di testata carica i dati esegue una query unica sulla tabella degli ordini e quindi le righe di ognuno degli ordini non sono caricate subito.



Esempio di videata master-detail

List detail

In questo secondo esempio vogliamo visualizzare nella parte sinistra della videata la lista dei prodotti, mentre nella parte destra tutti i dettagli del prodotto selezionato nella lista.



Esempio di videata list-detail

Anche in questo caso il coordinamento fra i due pannelli viene gestito dall'evento *onRowChanged* del pannello di lista, tramite il seguente codice:

```
$listPanel.onRowChanged = function(event)
{
  $detPanel.mainDM.document = event.newRow ? event.newRow.document : null;
};
```

In questo caso il documento attivo nel pannello di lista viene direttamente passato alla datamap principale del pannello di dettaglio impostando la proprietà *document*.

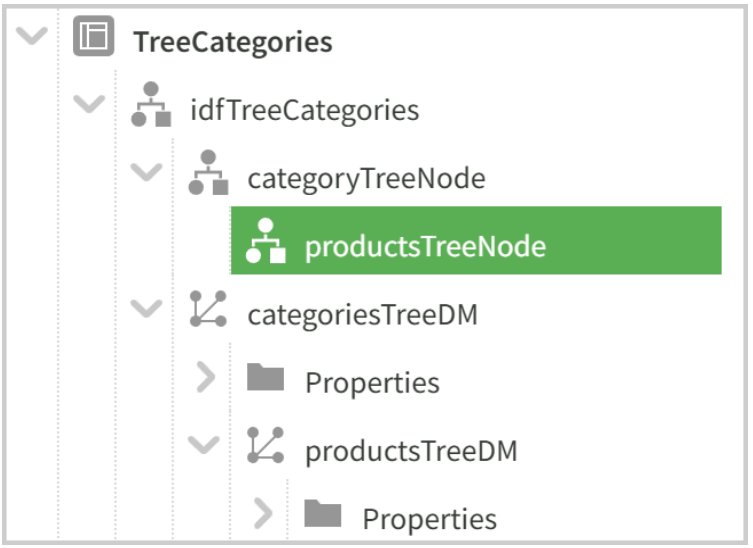
Coordinamento fra più di due pannelli

Utilizzando una combinazione delle tecniche precedenti è possibile coordinare anche più di due pannelli, anche se contenuti in videate diverse.

È importante notare che condividendo i documenti caricati in memoria fra più datamap si ottiene la sincronizzazione visuale delle variazioni senza dover effettuare query di refresh dei dati. È quindi il metodo di coordinamento da preferire.

Anatomia di un albero

La struttura degli elementi visuali che compongono un albero è mostrata nell'immagine seguente:

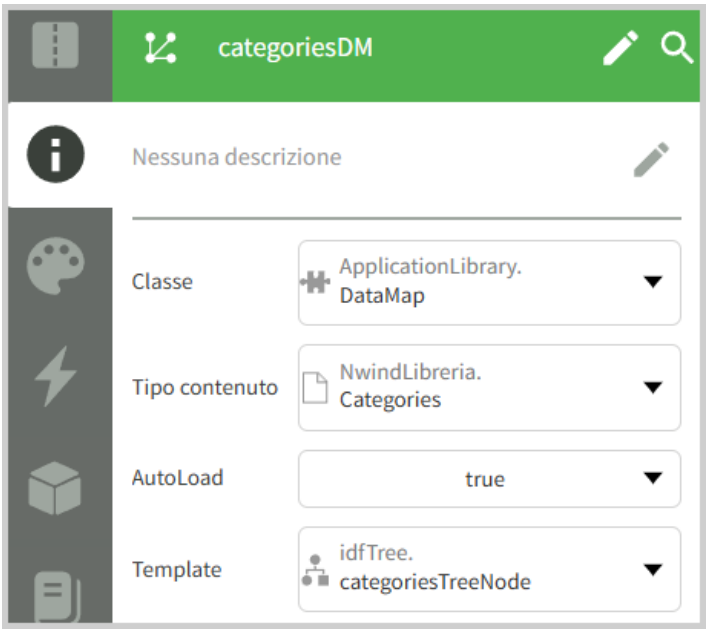


Il widget *IdfTree* può essere contenuto direttamente a livello base in una videata, se non sono previsti altri elementi, oppure può essere posizionato all'interno di qualunque altro elemento di tipo container, come ad esempio *IonContent* nel caso di pagina standard *Ionic*.

Facendo riferimento all'immagine precedente il primo elemento figlio dell'albero *idfTreeCategories* è *categoryTreeNode* un elemento di tipo *IdfTreeNode* (il nodo dell'albero) che rappresenta il primo livello dei dati. In esso verranno caricati i dati della datamap *categoriesTreeDM*.
Un nodo può contenere un altro nodo, come vediamo nell'immagine qui sopra dove sotto il nodo *categoryTreeNode* è presente il nodo *productTreeNode* che visualizza i dati caricati dalla datamap innestata *productsTreeDM*. Questa struttura permette di caricare al primo livello dell'albero degli elementi categoria e all'interno di ogni categoria gli elementi prodotto relativi.

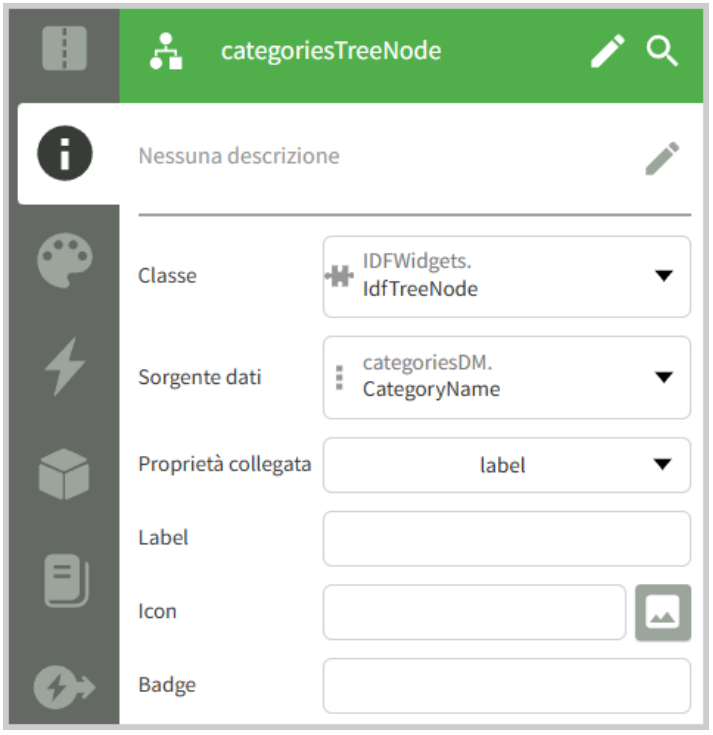
Dopo i nodi vediamo che sono presenti le datamap relative che forniscono ad ogni nodo i propri dati.

Aggiungendo un oggetto *IdfTreeNode* viene automaticamente aggiunta la datamap di caricamento dei dati il cui template è il nodo stesso come si vede nell'immagine più sotto. Il template rappresenta il modello visuale di un record della datamap quindi l'elemento che verrà ripetuto nella videata per ogni riga.



Un elemento IdfTreeNode può contenere altri elementi visuali che aggiungono funzionalità come bottoni, campi di input e immagini. Questo può essere utile per personalizzare il layout del nodo di un albero in modo da adeguarlo alle proprie esigenze ed uscire dallo schema standard di *label*, *icon* e *badge* per rappresentare un elemento, nel capitolo Layout personalizzato è illustrato un esempio esplicativo.

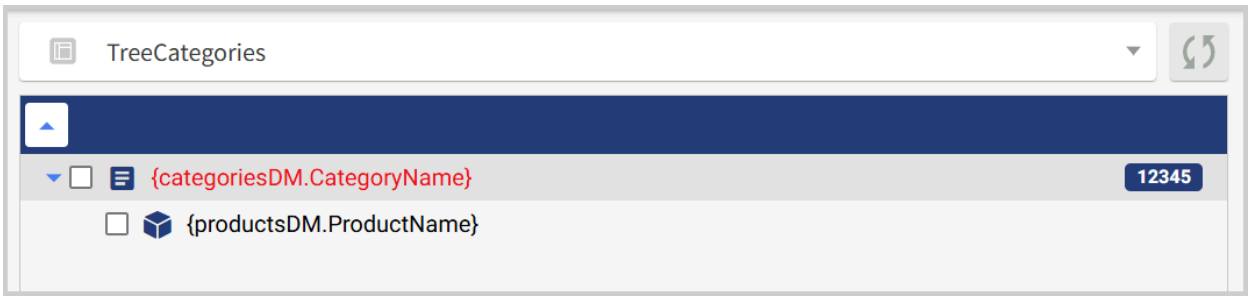
Nella barra delle proprietà del widget *IdfTreeNode* è possibile indicare quale campo del documento, sotteso alla datamap relativa, è la fonte dati per la *label* del nodo dell'albero nella proprietà *Sorgente dati*.



È possibile indicare un'icona per il nodo selezionandola tra quelle presenti nel progetto mediante la proprietà *icon*; oppure utilizzare un'immagine caricata nell'applicazione come risorsa di tipo immagine, facendo riferimento ad essa con *$nomeRisorsa*.
Inoltre è possibile utilizzare la proprietà *badge* che verrà visualizzato sul nodo come nell'immagine più sotto evidenziata nel riquadro rosso, questa proprietà è normalmente impostata a runtime.

I valori *label*, *icon* e *badge* possono essere valorizzati a runtime nell'evento *onRowComposition* della datamap sorgente dati del nodo.

Nell'editor delle videate l'albero appare come nell'esempio seguente:



La parte superiore dell'albero contiene la toolbar con il bottone per collassare l'elemento.
Nella toolbar dell'albero viene visualizzata la proprietà *Caption* dell'elemento *IdfTree*.

Nell'esempio qui sopra è abilitata la multi selezione degli elementi dell'albero che si attiva mediante la proprietà *ShowMultipleSelection*.

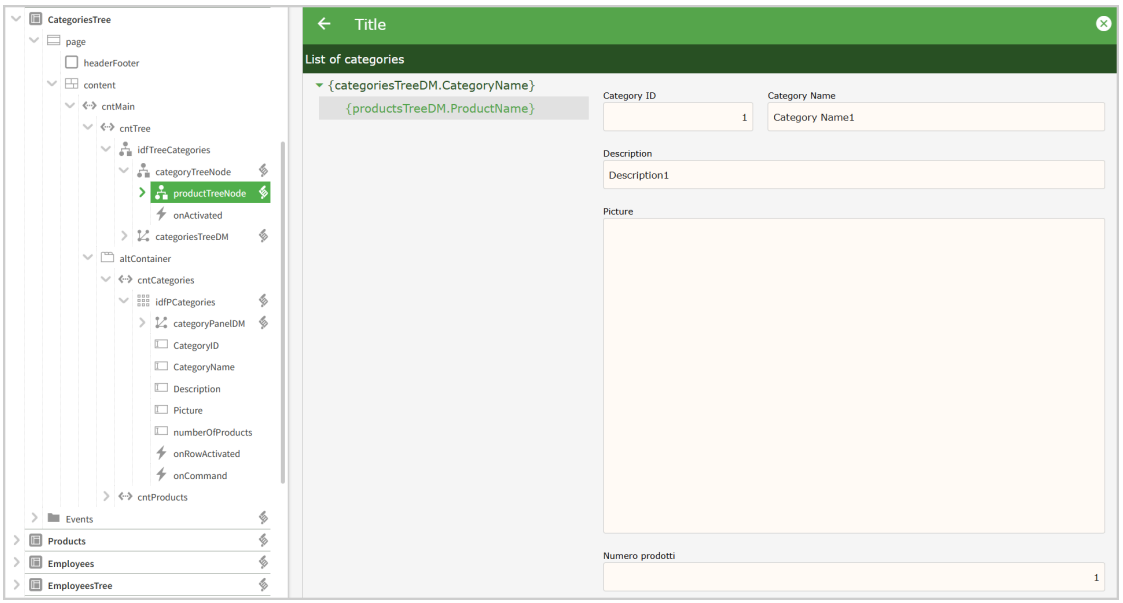
Creazione e modifica di un albero

Creazione di un albero

Per iniziare ad utilizzare gli alberi è necessario importare il pacchetto *FluidUI*, che contiene la libreria *IDFWidgets* con le definizioni degli elementi che compongono gli alberi.

A questo punto è possibile inserire il componente *IdfTree* sia a livello di videata che di elemento container. Mediante l'utilizzo dei container con layout *horizontal* o *vertical* è possibile combinare elementi alberi e pannelli nella stessa videata.

Nell'immagine seguente possiamo vedere una videata con al suo interno un albero e dei pannelli in un layout composto da container e altContainer.

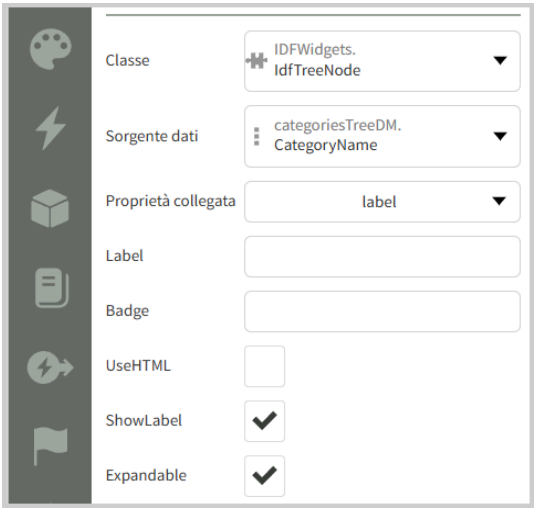


Albero affiancato ad un pannello mediante un container a layout horizontal

Subito dopo aver inserito nella videata il componente *IdfTree*, al quale automaticamente sono aggiunti un elemento *IdfTreeNode* e una datamap, viene selezionata la datamap del nodo. A questo punto è possibile indicare il tipo di documenti che verranno mostrati nel nodo dell'albero.

Come per ogni altra datamap è possibile utilizzare una query di caricamento dei dati, oppure specificare le proprietà e caricare i dati direttamente in memoria. La soluzione migliore e più flessibile è sempre quella di passare attraverso i documenti.

Per definire quale informazione visualizzare nella *label* del nodo dell'albero si imposta la proprietà *Sorgente dati* con il relativo campo della datamap come da immagine seguente.



L'informazione contenuta nella proprietà *label* può anche contenere codice HTML e in questo caso per farla interpretare correttamente dal framework occorre impostare a true la proprietà *useHTML* del nodo.

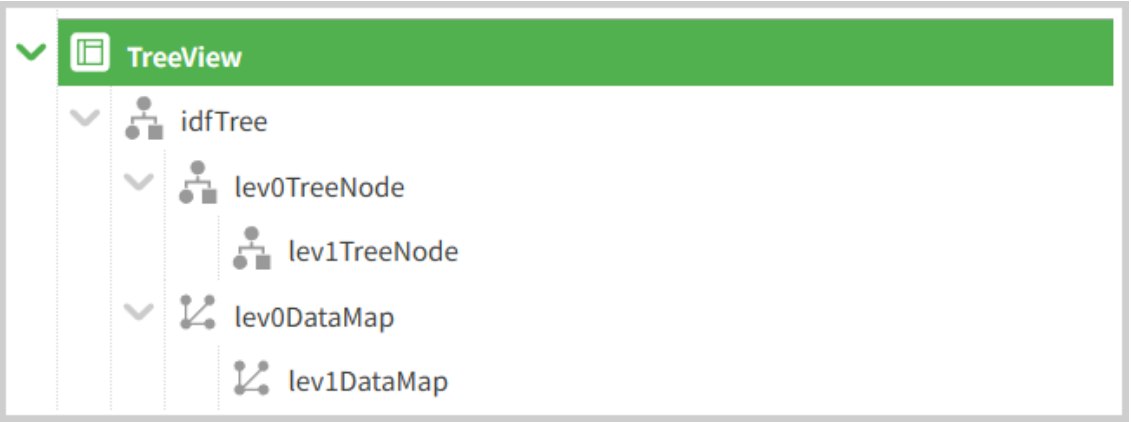
Oltre alla proprietà *label* esiste anche una proprietà *badge* che è possibile impostare a design time oppure a runtime utilizzando l'evento *onRowComposition* della datamap del nodo.
Nel codice qui sotto impostiamo il *badge* del nodo con il numero di prodotti contenuti in una categoria.

```
$categoriesTreeDM.onRowComposition = function(row, template)
{
  $categoryTreeNode.badge = row.numberOfProducts;
};
```

È possibile attivare una visualizzazione compatta di un albero, in modo che occupi meno spazio in altezza sui nodi, impostando a true la proprietà *compacted* dell'elemento *IdfTree*.

Modifica di un albero

Per aggiungere un ulteriore livello ad un albero è sufficiente inserire, sotto al nodo interessato, un altro elemento *IdfTreeNode*. L'aggiunta di un nodo sotto un altro nodo determina l'inserimento di una nuova datamap innestata sotto quella del livello precedente dell'albero come si vede nell'immagine seguente.



In questo modo è possibile realizzare strutture complesse a piacimento utilizzando elementi *IdfTreeNode* nidificati.

La proprietà *label* dei nodi dell'albero può essere personalizzata utilizzando gli stili inline degli oggetti *IdfTree* e *IdfTreeNode* per definire grassetto, colore di sfondo e tutte le altre proprietà di stile di un elemento visuale.

Naturalmente possono essere personalizzate anche le classi CSS dell'elemento *IdfTree* che è possibile trovare nella risorsa *treeCSS* della classe *IdfTree* della libreria *IDFWidgets*.

Al suo interno possiamo trovare la definizione di default delle variabili colore dei vari elementi dell'albero e tutte le classi CSS che lo configurano; Qui di seguito è riportato un esempio delle variabili di configurazione:

```
:root {
  /* Toolbar */
  --col-back-primary-tree: color-mix(in srgb, $primary 50%, black);
  --col-fore-primary-tree: #ffffff;

  /* Toolbar buttons, icons */
  --col-back-secondary-tree: #ffffff;
  --col-fore-secondary-tree: $primary;

  --col-tree-back: #f8f8f8;

  /***** Nodes *****/
  --col-active-node-back: #e5e5e5;
  --col-node-hover: #eeeeee;

  --node-height: 32px;
  --compact-node-height: 24px;

  --col-focused-node-border: #d0d0d0;
}
```

Se per esempio vogliamo modificare l'altezza di tutti i nodi nella nostra applicazione possiamo semplicemente sovrascrivere la variabile *--node-height* nel CSS della nostra applicazione:

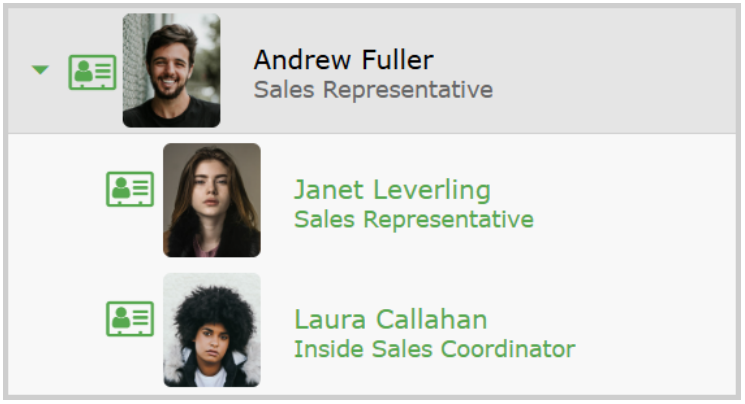
```
:root {
  --node-height: 32px !important;
}
```

È necessario indicare la clausola *!important* per garantire la sovrascrittura della regola.

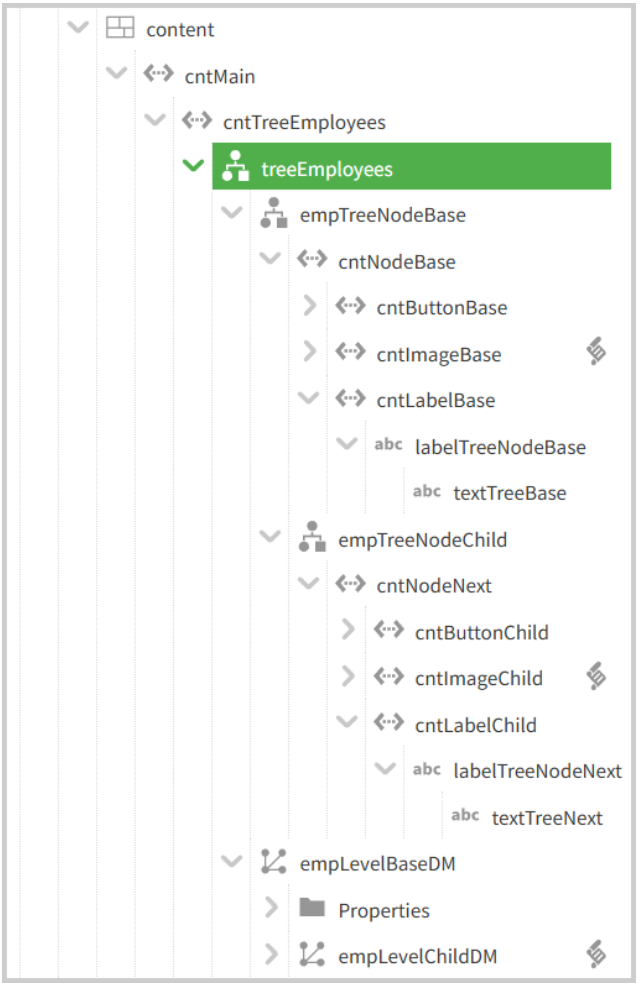
Layout personalizzato

In alternativa all'utilizzo della *label* per visualizzare il contenuto di un nodo è possibile creare un proprio layout. Per disabilitare la visualizzazione della *label* di default occorre impostare a false la proprietà *showLabel*.

Per esempio se si vuole realizzare una struttura come quella nell'immagine seguente, dove abbiamo che un nodo è composto da: un bottone, un'immagine e due testi uno sotto l'altro, occorre procedere come descritto più in basso.



Per prima cosa si disabilita l'utilizzo della *label* di default mediante la proprietà *Show label*. Quindi si deve realizzare una struttura come quella che vediamo nell'immagine seguente.



Qui, sotto l'elemento *empTreeNodeBase*, abbiamo il container *cntNodeBase* con layout *horizontal* che a sua volta ha tre container:

- *cntButtonBase* - che contiene un *IonButtons* con al suo interno un *IonButton*
- *cntImageBase* - che contiene un *Image*
- *cntLabelBase* - che contiene un *IonLabel* con al suo interno un *IonText*

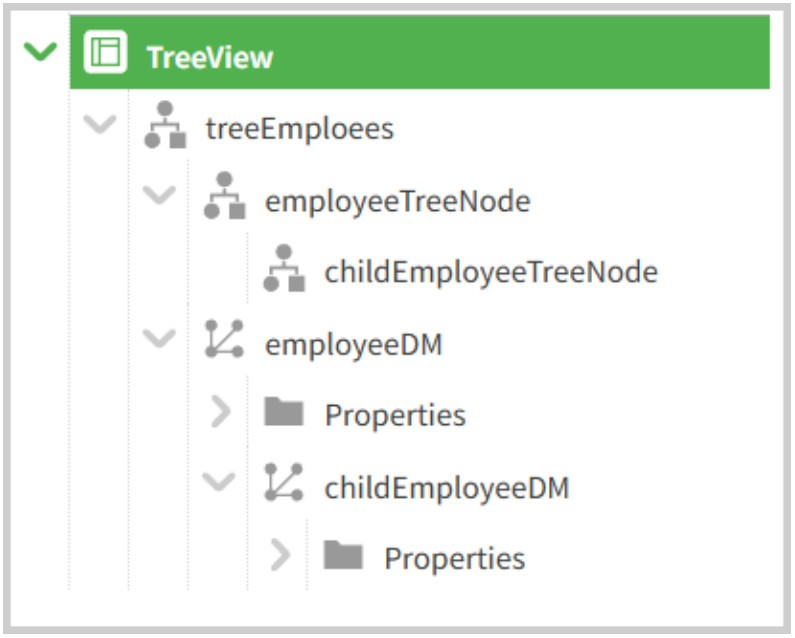
A questi tre container è possibile associare l'evento *onClick* per far compiere specifiche azioni ad ognuno di essi. All'interno dell'evento è possibile fare riferimento ai dati della datamap mediante *this.row*.

In questo esempio è stata modificata l'altezza del nodo impostando *height: 80px* negli stili inline dell'elemento *empTreeNodeBase* per adeguare lo spazio alle dimensioni dell'immagine.

Alberi ricorsivi

Per realizzare un albero ricorsivo, che naviga una struttura gerarchica di una tabella con legami padre figlio di cui non sappiamo in partenza il livello di profondità, dovremo utilizzare le datamap innestate ricorsive (vedere il manuale Datamap al capitolo Datamap ricorsive).

Se vogliamo realizzare l'albero della struttura dei legami tra i record della tabella *Employees*, che ha il campo *ReportsTo* che contiene l'id del proprio record padre, dobbiamo utilizzare la seguente struttura.



La datamap *employeeDM* ha come *Tipo contenuto* il documento *Employee*. Questo documento ha una collection figlia di documenti di tipo Employee legati dalla foreign key sul campo *ReportsTo* (il campo che realizza la relazione padre figlio).
La datamap *childEmployeeDM* ha come *Tipo contenuto* la collection figlia del documento *Employee* in questo modo sa esattamente quali record caricare e con quale chiave.
Sulla datamap childEmployeeDM è impostata la proprietà *Recursive* a true.

La datamap di livello zero *employeeDM* che alimenta il nodo *employeeTreeNode* ha un filtro di caricamento che prende tutti i record della tabella *Employees* senza un padre (quelli con il campo *ReportsTo* vuoto). Questo lo realizziamo impostando il codice seguente nell'evento *onLoad* della videata.

```
$employeeDM.setFilter(App.NwindBE.Employee.ReportsTo, "!");
yield $employeeDM.load();
```

Una struttura ricorsiva di questo tipo viene creata automaticamente dall'ide di Instant Developer Cloud quando il documento utilizzato nella datamap del nodo ha una collection figlia di documenti del suo stesso tipo unica cosa da fare è scrivere il codice che filtra il livello base dell'albero come abbiamo visto poco sopra.

Gli alberi a runtime

In questo paragrafo vediamo le dinamiche dell'albero per controllare il suo comportamento a runtime tramite eventi e metodi. Alcuni comportamenti verranno gestiti mediante la datamap del nodo dell'albero che ne coordina l'accesso ai dati.

Un esempio di utilizzo degli alberi è nel progetto di esempio North Wind Web che è possibile trovare nella sezione *Documentazione* della Console di Instant Developer Cloud.

Quando un albero viene aperto è possibile gestire manualmente o automaticamente il caricamento dei dati agendo sulla datamap legata al nodo di livello base.
La proprietà *autoload* della datamap per default è settata a true ma è possibile impostarla a false e poi gestire il caricamento dei dati nell'evento *onLoad* della videata dopo aver indicato gli opportuni filtri.

Configurazione della visualizzazione dei nodi

In molti casi è necessario modificare lo stile di un singolo nodo dell'albero in funzione del contenuto del record collegato, a tal fine è possibile utilizzare l'evento *onRowComposition* della datamap di caricamento del nodo.

Se per esempio volessimo colorare di rosso la *label* dei nodi dei prodotti sotto scorta di un albero che li visualizza potremmo utilizzare il seguente codice:

```
$EmployeeDM.onRowComposition = function(row, template)
{
   $productTreeNode.style.color =
     (row.UnitsInStock < row.ReorderLevel) ? "red" : "";
};
```

Accesso ai dati del nodo attivo

Se si desidera accedere ai dati che alimentano il nodo di un albero è possibile recuperare il documento sotteso dalla datamap facendo riferimento a *this.row.document* nel caso la datamap sia basata sui documenti.

Altrimenti per un albero con datamap che dipende da una query a da proprietà in memoria si recuperano i dati della riga attiva mediante *this.row.nome-proprietà*, dove *nome-proprietà* è il nome del campo che si vuole utilizzare così come riportato nella query.

Se si vuol accedere ai dati di un nodo dell'albero quando esso viene attivato e impostare la caption dell'albero della toolbar con la *label* del nodo attivo è possibile utilizzare l'evento *onNodeActivated* dell'albero.

```
$idfTree.onNodeActivated = function(event)
{
  this.caption = event.treeNode.label;
};
```

Nel caso venga utilizzato l'evento *onActivated* del nodo invece che quello dell'albero faremo riferimento a *this.label* per la *label* del nodo e a *this.row.document* per il documento relativo.

Al momento del caricamento dei dati di un albero tramite la sua datamap occorre sapere che non esiste un nodo attivo e quindi se vogliamo attivare il primo nodo dell'albero alla sua apertura lo dobbiamo fare utilizzando questo codice nell'evento *onLoad* della videata.

```
App.CategoriesTree.prototype.onLoad = function(options)
{
  yield $idfTreeDatamap.load();
  $idfTreeDatamap.position = 0;
  let doc = $idfTreeDatamap.row.document;
  let node = yield $idfTree.findNode({CategoryID : doc.CategoryID});
  node.activate();
};
```

Nell'esempio viene caricata la datamap che alimenta il nodo, attivata la prima riga della datamap, recuperato il documento relativo della posizione, recuperata l'istanza del nodo e su questa richiamato il metodo che lo attiva.

Il metodo *findNode* permette di cercare un nodo in base a svariati parametri che è possibile visualizzare nella reference all'interno dell'ide di Instant Developer Cloud premendo F1 una volta selezionato il metodo nell'albero del progetto.
Nella libreria *IDFWidgets* è possibile visualizzare tutti i metodi ed eventi degli elementi IdfTree e IdfTreeNode per una più esaustiva lista delle possibilità di manipolazione di questi oggetti.

La proprietà *activeNode* di un albero restituisce il nodo attivo dell'albero e può essere utilizzata per recuperarne i dati e utilizzarli in altre parti di una videata.

Selezione multipla

Gli alberi supportano la selezione multipla dei dati e tale opzione è attivabile impostando a true la proprietà *ShowMultipleSelection*.

I nodi dell'albero selezionati sono controllabili tramite interfaccia utente o da codice. In questo secondo caso è possibile usare il metodo, di albero o di nodo, *changeAllNodesSelection* per cambiare la selezione di tutti i nodi.
La proprietà selected del nodo indica se quel nodo è selezionato o meno e può essere modificata per cambiarne la selezione.

Per permettere di personalizzare la modifica della selezione l'albero notifica l'evento *onNodeSelectionChanged* e il singolo nodo notifica l'evento *onSelectionChanged* tutte le volte che un nodo cambia stato di selezione oppure se è stato usato un comando di cambiamento della selezione complessivo.

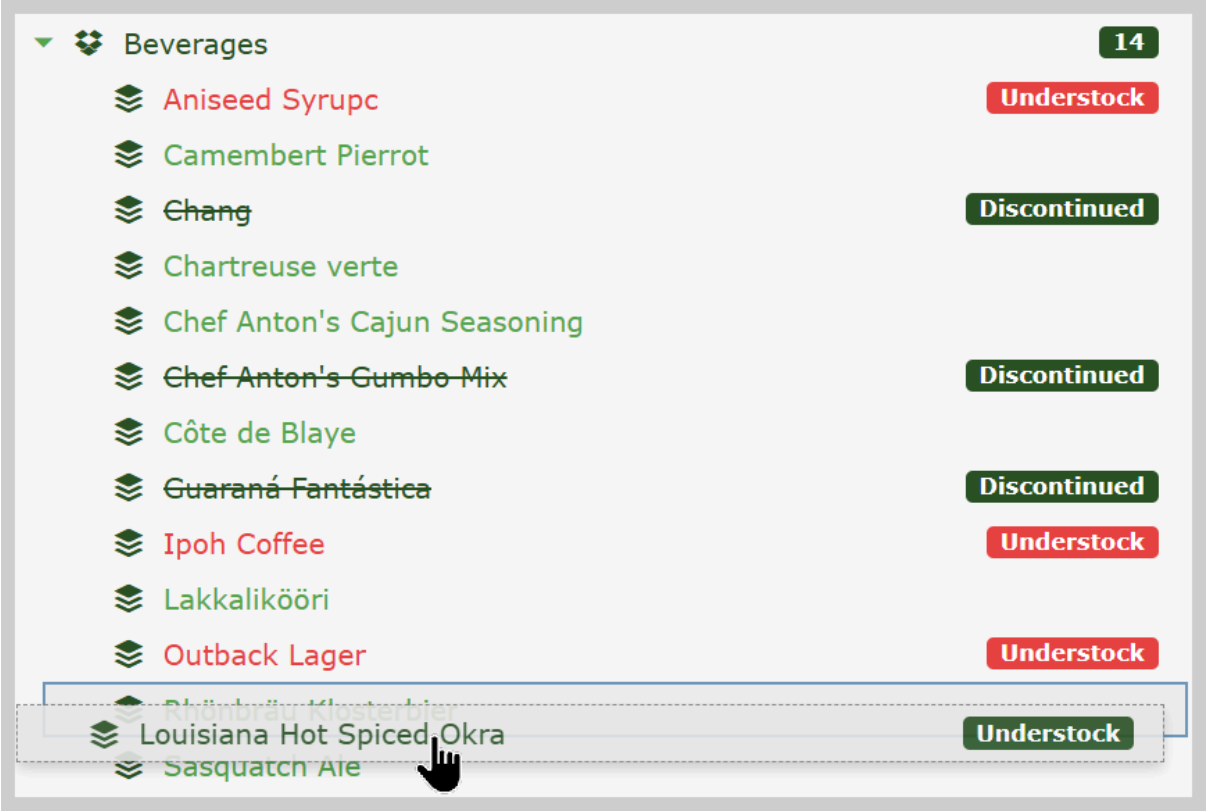
Nel caso si debbano eseguire operazioni sui nodi selezionati di un albero è possibile utilizzare il metodo *getSelectedNodes* dell'albero che restituisce un array contenente i nodi selezionati.

Drag & drop

In un albero è possibile attivare l'utilizzo del drag & drop, per spostare un nodo su un altro nodo, mediante l'impostazione a true delle proprietà *CanDrag* e *CanDrop* dell'elemento *IdfTree*.

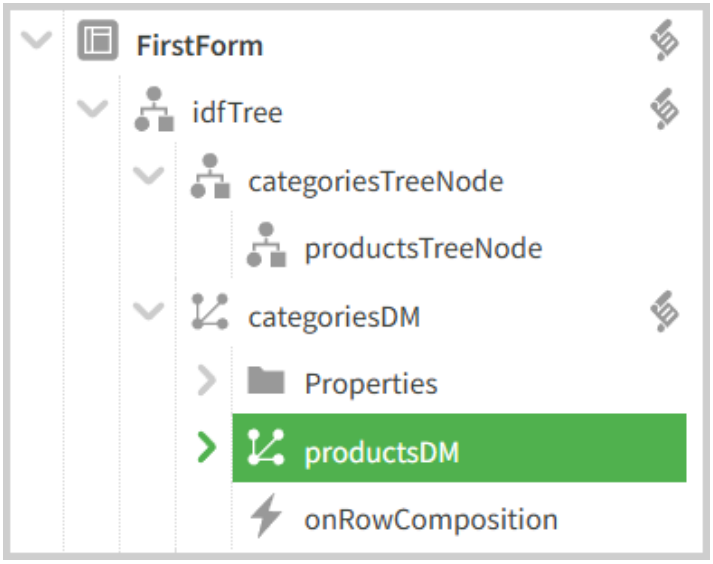
Sull'elemento nodo abbiamo la proprietà *Draggable* che indica che esso è trascinabile su altri elementi, e la proprietà *Droppable* che indica se è possibile trascinare elementi su di esso.

Nell'immagine sottostante possiamo vedere un drag & drop di un prodotto nell'albero delle categorie.



Le operazioni di drag & drop gestiscono automaticamente lo spostamento del nodo dalla collection di partenza a quella di destinazione.
Non occorre quindi eseguire operazioni specifiche per avere le strutture delle datamap aggiornate e se esse dipendono da documenti è sufficiente utilizzare il metodo save della datamap per aggiornare i dati e rendere effettivo lo spostamento.

Quindi se abbiamo un documento *Category* che ha una collection figlia *Products* di documenti *Product* e su questa struttura costruiamo un albero avremo la situazione dell'immagine sottostante.



L'albero *idfTree* ha le proprietà *canDrag* e *canDrop* impostate a true, il nodo *categoriesTreeNode* ha la proprietà *Droppable* impostata a true e il nodo *productsTreeNode* ha le proprietà *Draggable* e *Droppable* impostate a true. Quindi è possibile trascinare nodi prodotto su nodi categoria e altri nodi prodotto.

Trascinando un nodo prodotto su un nodo categoria avremo come risultato che la collection Products della categoria di destinazione conterrà il prodotto trascinato in ultima posizione.

Se invece si trascina un nodo prodotto su un altro nodo prodotto si avrà l'effetto di aver spostato quest'ultimo di posizione nella collection relativa della datamap.

Una volta effettuato il drag & drop è possibile eseguire il salvataggio del documento spostato per consolidare la modifica sul database. Tale operazione si può realizzare mediante apposito bottone nell'interfaccia utente oppure utilizzando l'evento *onDataChange* della datamap.

Nell'evento quindi possiamo controllare se la datamap è in stato modificato ed effettuare un salvataggio della stessa così da riportare le modifiche sul database.

```
$categoriesDM.onDataChange = function()
{
  if ($categoriesDM.modified) {
    yield $categoriesDM.save();
  }
};
```

Quando invece spostiamo un documento *Product* su un altro documento *Product* della stessa categoria questo cambia posizione all'interno della datamap *Products* e quindi è

possibile aggiornare un eventuale campo sequenza che ne determina l'ordine di visualizzazione.

Per interagire con le operazioni di drag & drop ed eventualmente annularle si utilizza l'evento *onDrop*, presente sia a livello di albero che di nodo.
L'evento *onDrop* dell'albero riceve come parametri sia il nodo trascinato che quello su cui è stato rilasciato, mentre l'evento *onDrop* del nodo riceve come parametro l'elemento che è stato trascinato.

Nel caso di evento *onDrop* sull'albero per il trascinamento di un nodo del documento *Product* su un nodo del documento *Category* potremmo scrivere il seguente codice per impedire lo spostamento di un prodotto sulla categoria con id uguale a 1.

```
$idfTreeCategories.onDrop = function(event)
{
  if (event.dragElement.row instanceof App.NwindBE.Product &&
      event.dropElement.row instanceof App.NwindBE.Category) {
    let nodeDrag = event.dragElement.row; //type:NwindBE.Product
    let nodeDrop = event.dropElement.row; //type:NwindBE.Category
    //
    if (nodeDrop.CategoryID === 1) {
      event.skip = true;
      yield app.popup({
        type : "toast",
        message : "Spostamento non ammesso"
      });
    }
  }
};
```

Nell'esempio viene controllato che il nodo di provenienza sia di un documento di tipo *Product* e quello di destinazione sia di tipo *Category*, quindi sono recuperati i relativi documenti e se la categoria di destinazione è la 1 viene annullata l'operazione e visualizzato un messaggio.

Navigazione tramite interfaccia utente

L'albero visualizzato in una videata è navigabile da tastiera in modo che l'utilizzatore possa spostarsi tra i suoi nodi mediante i tasti freccia, espandere o contrarre nodi dell'albero e attivare un nodo senza dover usare il mouse.

Ecco un elenco dei comandi di navigazione possibili:

- freccia su e giù - spostamento tra i nodi dell'albero
- freccia destra - espansione del nodo corrente, se il nodo è già espanso sposta la selezione sul primo nodo figlio
- freccia sinistra - contrazione del nodo corrente, se il nodo selezionato è un figlio la selezione torna al nodo padre
- enter - attivazione del nodo corrente
- barra spaziatrice - nel caso di multi selezione attiva, seleziona o deseleziona il nodo