Versione 22.0 - Note di rilascio

Novità

Questa sezione contiene le modifiche più importanti o che implicano modifiche necessarie al comportamento del framework. Si consiglia di prendere in considerazione questo paragrafo prima di installare questa versione sui propri server IDE o di produzione.

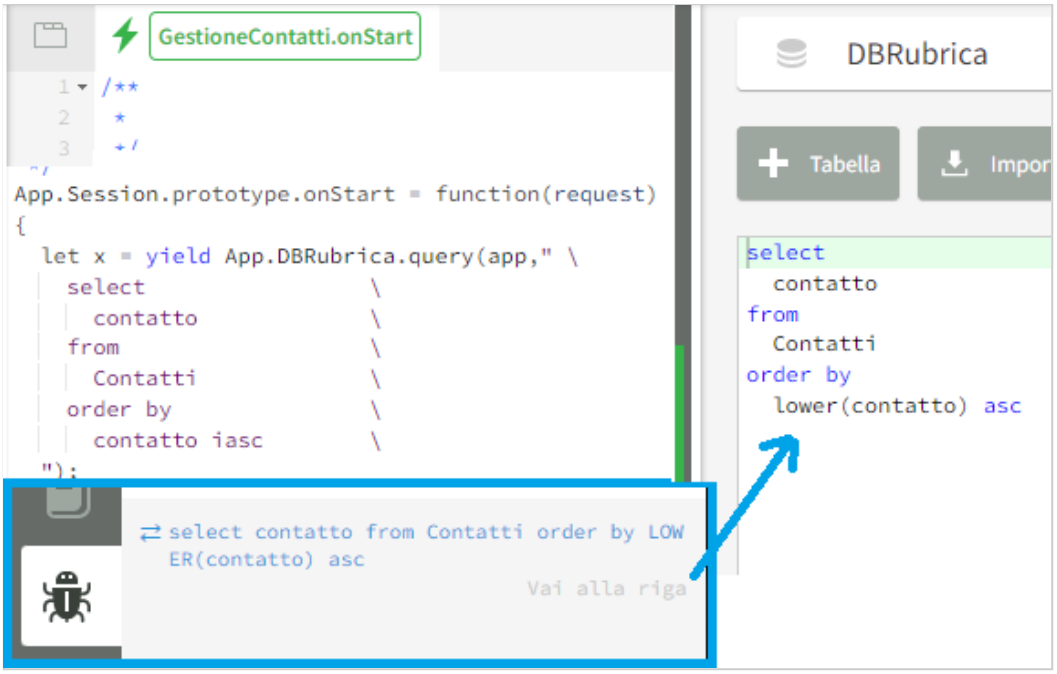
Formattazione del codice

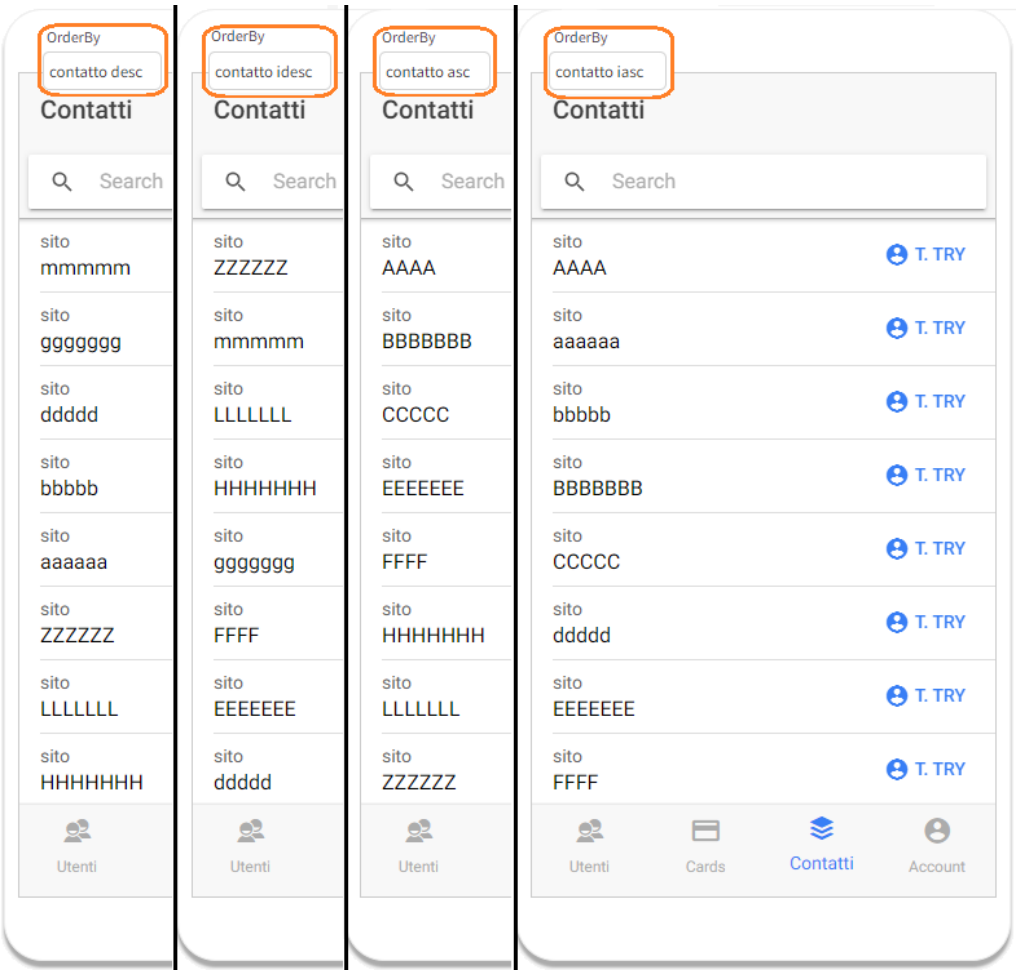
All'editor di codice è stata aggiunta la possibilità di formattare il codice. Per attivare questa funzione è sufficiente utilizzare la voce di menu contestuale *Formatta codice*.

Ordinamento case-insensitive

È ora possibile ordinare le datamap o le query strutturate anche in modo *case insensitive*. Per farlo è sufficiente utilizzare i nuovi attributi *iasc* e *idesc* nelle proprietà *orderBy* e *groupBy* delle datamap o nelle clausole *order by* delle query.

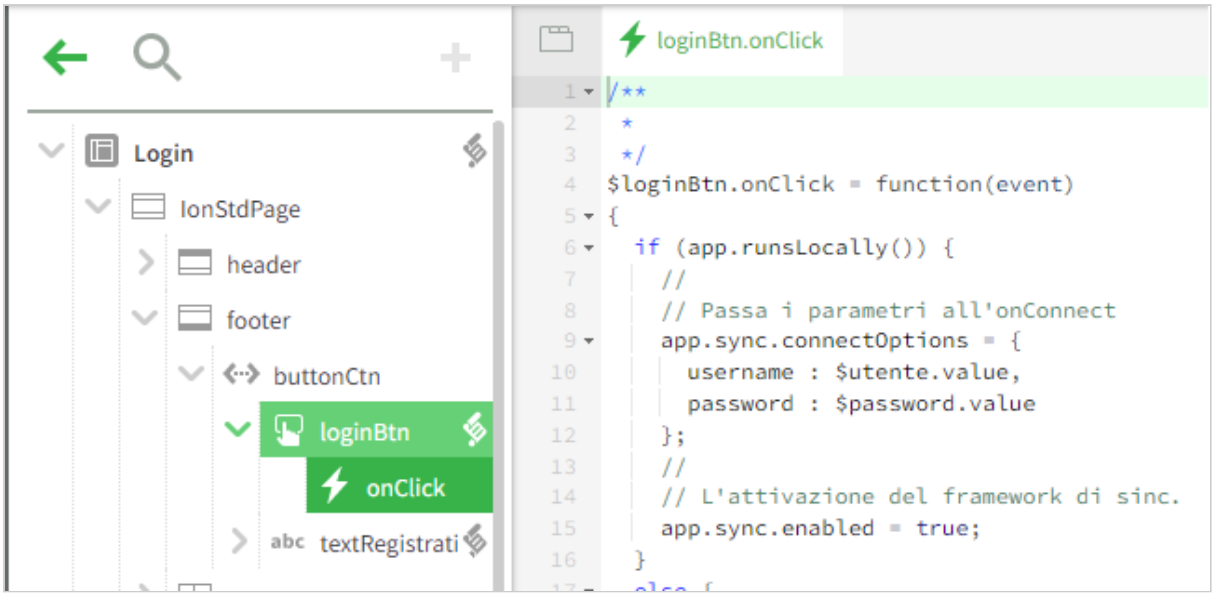
In fase di esecuzione della query, il framework sostituisce l'espressione *nomeCampo iasc* con *lower(nomeCampo) asc* e *nomeCampo idesc* con *lower(nomeCampo) desc*.


GestioneContatti.onStart
/**
*
*/
App.Session.prototype.onStart = function(request)
{
let x = yield App.DBRubrica.query(app," \
select \
contatto \
from \
Contatti \
order by \
contatto iasc \
");
select contatto from Contatti order by LOWER(contatto) asc
Vai alla riga
DBRubrica
Tabella
Impor
select
contatto
from
Contatti
order by
lower(contatto) asc

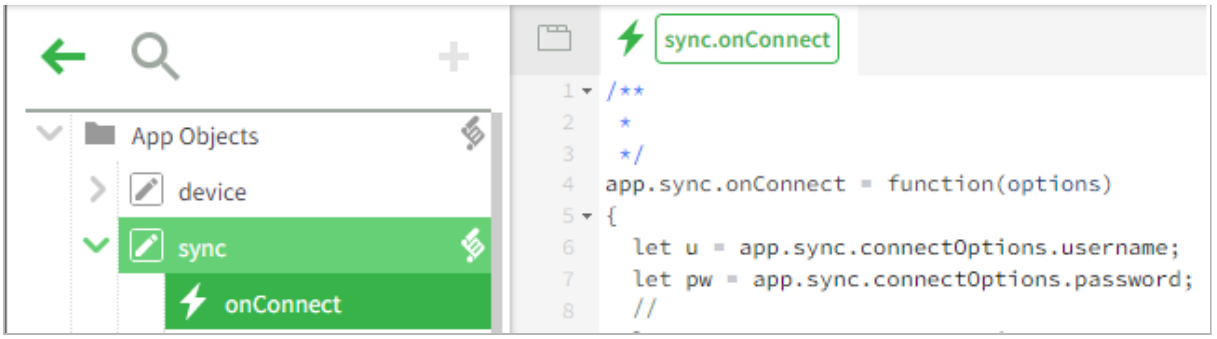

OrderBy
contatto desc
Contatti
Search
sito mmmmm
sito ggggggg
sito ddddd
sito bbbbb
sito aaaaaa
sito ZZZZZZ
sito LLLLLLL
sito HHHHHHH
Utenti
OrderBy
contatto idesc
Contatti
Search
sito ZZZZZZ
sito mmmmm
sito LLLLLLL
sito HHHHHHH
sito ggggggg
sito FFFF
sito EEEEEEE
sito ddddd
Utenti
OrderBy
contatto asc
Contatti
Search
sito AAAA
sito BBBBBBB
sito CCCCC
sito EEEEEEE
sito FFFF
sito HHHHHHH
sito LLLLLLL
sito ZZZZZZ
Utenti
OrderBy
contatto iasc
Contatti
Search
sito AAAA
sito aaaaaa
sito bbbbb
sito BBBBBBB
sito CCCCC
sito ddddd
sito EEEEEEE
sito FFFF
T. TRY
Utenti
Cards
Contatti
Account

Nuova proprietà app.sync.connectOptions

È stata aggiunta la proprietà *app.sync.connectOptions*, utile per inviare informazioni al server in fase di apertura della connessione.
Questa proprietà di tipo oggetto viene passata alla controparte al momento dell'apertura della connessione e permette quindi di inviare dati senza dover ricorrere alla proprietà *topics*. È quindi il metodo migliore per inviare le credenziali di accesso al server. I dati arrivati dal client sono già disponibili nell'evento *app.sync.onConnect* della sessione proxy.



Apertura della connessione lato client: uso di connectOptions per comunicare le credenziali



Apertura della connessione lato server: lettura di connectOptions per verificare le credenziali

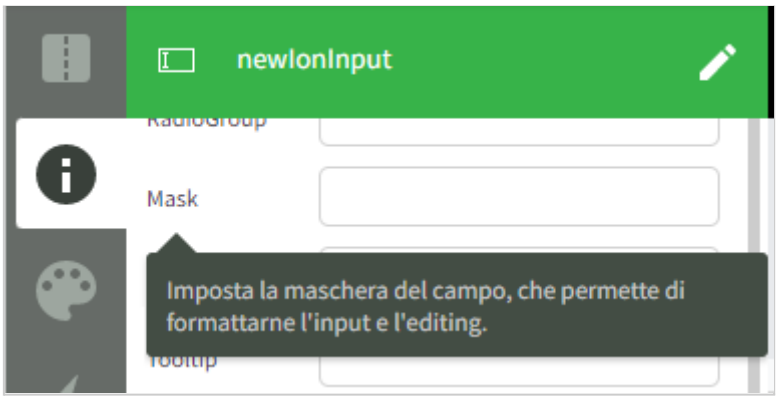
Per maggiori informazioni riguardo all'evento *app.sync.onConnect*, è possibile consultare il manuale 07-Sincronizzazione.

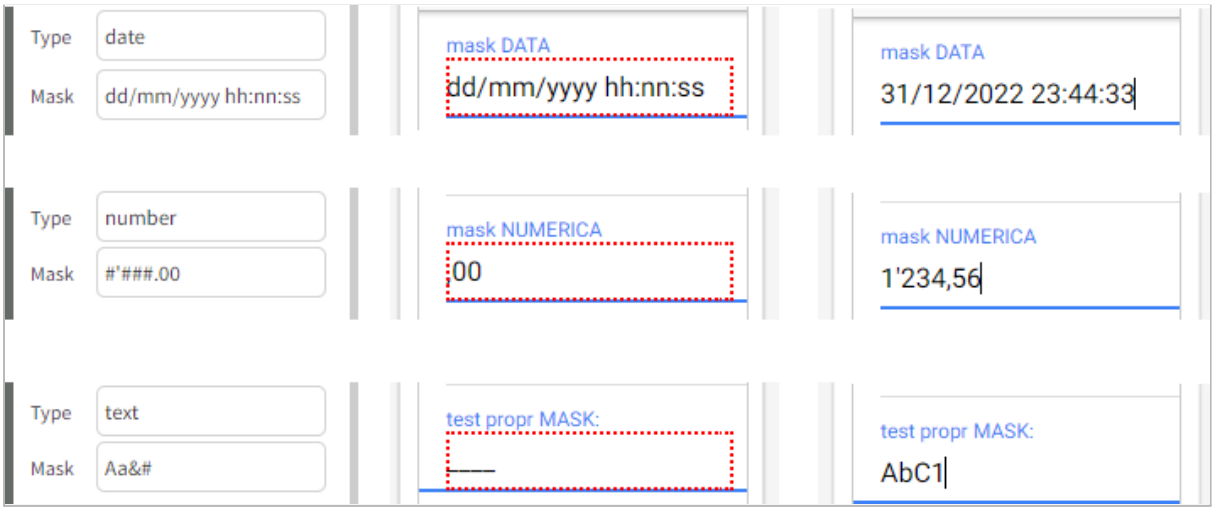
Gestione maschera per elementi visuali di tipo input

Ora gli elementi visuali derivati da *App.Input* possono gestire la maschera di inserimento dati. A tal fine è stata aggiunta la proprietà *mask* che permette di definirla. La maschera viene utilizzata sia per mostrare il valore che per guidarne l'inserimento da parte dell'utente.

La sintassi della maschera dipende anche dal tipo di input, secondo le seguenti regole:

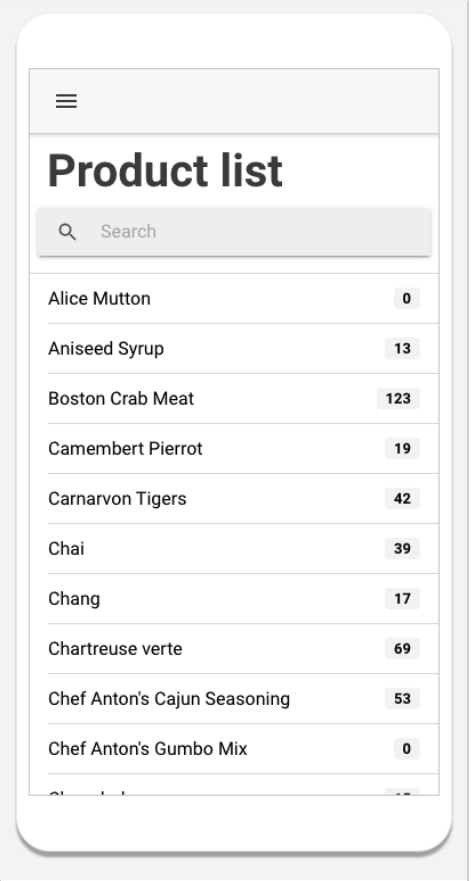
- Input di tipo data / data ora: la proprietà *type* deve essere impostata a *date*; *mask* è una stringa composta dalla combinazione dei token *yy yyyy mm dd hh nn ss.*
- Input di tipo numerico: *type* deve essere impostata a *number*, *mask* è una stringa composta dalla combinazione dei caratteri *# (cancelletto: cifra opzionale) . (punto: separatore decimale) , (virgola: raggruppamento migliaia) 0 (zero: cifra non opzionale)*.
- Input di tipo carattere: *type* deve essere *text* o vuoto, *mask* è una stringa composta dalla combinazione dei caratteri *A (lettera maiuscola) a (lettera minuscola) & (lettera maiuscola o numero) # (numero).*





Intestazione videate in evidenza

Ora è possibile configurare gli elementi base di una videata per ottenere un'intestazione evidenziata che, al momento dello scroll del contenuto, torna ad essere un'intestazione normale. L'effetto è mostrato nell'immagine seguente:



Per ottenere questo risultato occorre impostare la nuova proprietà *showTitleContent* dell'elemento *IonNavBar* ad uno dei seguenti valori:

- *disabled* (valore di default): l'intestazione si comporta come nelle versioni precedenti.
- *enabled*: viene attivato l'effetto di evidenziazione.
- *auto*: l'effetto viene attivato nei dispositivi iOS ma non su Android o Web.

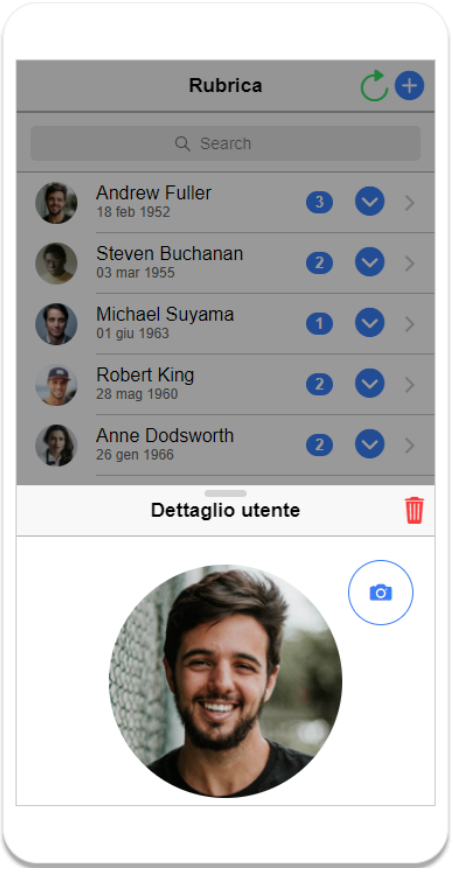
Questa proprietà richiede che la pagina abbia la seguente struttura:

- IonPage -> IonHeader -> IonNavBar -> IonTitle
- IonPage -> IonHeader -> IonToolbar -> IonSearchbar (elementi opzionali)
- IonPage -> IonContent

Questa funzionalità gestisce anche un elemento IonToolbar inserito subito dopo IonNavBar che può contenere una IonSearchbar. Tale elemento viene spostato nel content, subito dopo il titolo.

Apertura parziale di videate popup modali

È ora possibile visualizzare videate popup modali su smartphone in modo da occupare solo parzialmente lo schermo. La videata può essere trascinata con il dito in modo da essere ulteriormente espansa, ridotta o dismessa.



Per controllare questo effetto è stata aggiunta la proprietà *height* nelle opzioni di apertura della videata, come esemplificato dalla seguente riga di codice:

```
App.Pages.push( App.DetailForm, {
        popup:true, modal:true, height: ["20%", "50%", "100%"]
});
```

La proprietà di apertura *height* può essere impostata ad un valore singolo o ad un array di valori. Nel secondo caso, l'utente potrà trascinare la barra del titolo per espandere o ridurre la videata. I valori possono essere espressi come stringhe in pixel oppure in percentuale.

Questa proprietà ha l'effetto indicato solo su device smartphone e per videate modali. Le opzioni *popup:true* e *modal:true* sono quindi necessarie.

Gestione delle dimensioni dei popup su tablet e desktop

Per poter gestire più semplicemente le dimensioni dei popup anche su tablet e desktop, sono state aggiunte le proprietà di apertura *width* e *height* che rappresentano le dimensioni in pixel o in percentuale della videata popup.

Controllo automatico dello spazio su disco

Per evitare rischi di blocco di server a causa del riempimento dei dischi dati, sono stati aggiunti dei sistemi automatici di controllo e gestione del problema.

Spazio sul disco dati del server di produzione

I server di produzione effettuano un controllo periodico, ogni minuto, dello spazio ancora a disposizione sul loro disco dati, su cui sono installate le applicazioni.

Se lo spazio scende sotto i 500 MB, sia il log di sistema che il log strutturato delle applicazioni vengono interrotti. Viene aggiunto un ultimo messaggio di warning *App log has been stopped due to low free disk space* per avvisare della situazione.

Per riattivare il log è sufficiente eliminare i file non necessari dal server e attendere il prossimo controllo eseguito dal sistema.

Per gestire lo spazio occupato nel disco dati del server, è possibile utilizzare:

- La pagine di esplorazione del file system delle applicazioni nella console.
- L'elenco dei database presenti sul server che ne mostra la relativa dimensione.
- La pagine di gestione delle sessioni del log strutturato per cancellare quelle meno recenti.

Per maggiori informazioni riguardo al log strutturato è possibile consultare il manuale 17-I server di produzione.

Dimensione massima del log strutturato per singola sessione

Se il log strutturato di una singola sessione supera i 50 MB, il sistema interrompe il log per quella sessione e genera il messaggio di warning *App log has been stopped because the log file is too big (> 50 MB).*

Controllo spazio dei server IDE

Se lo spazio libero dei disco dati di un server IDE scende sotto i 250 MB, sarà possibile aprire i progetti unicamente in sola lettura.

Per poter modificare nuovamente i progetti, è necessario recuperare spazio disco eliminando i file non necessari, tramite i seguenti strumenti:

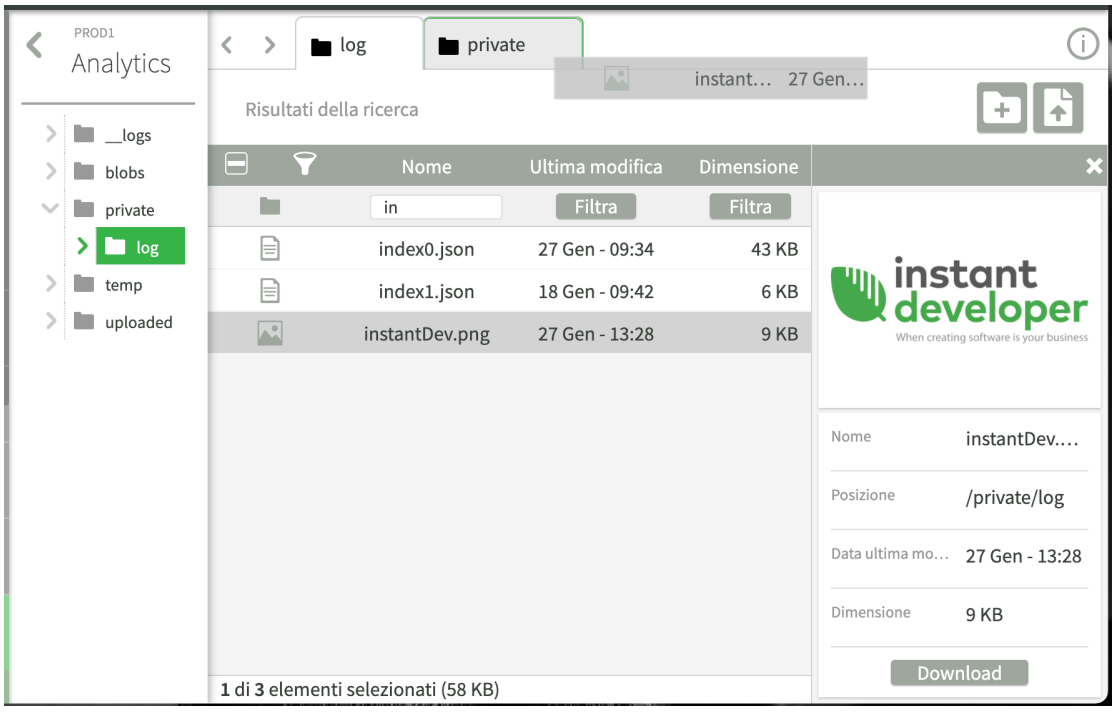
- La pagine di esplorazione del file system delle applicazioni nella console.
- L'elenco dei database presenti sul server che ne mostra la relativa dimensione.
- L'elenco delle build disponibili per ogni applicazione in cui è possibile cancellare quelle non più utilizzate.
- La pagina di dettaglio di un progetto che mostra lo spazio occupato dal sistema di teamworking.

Nuovo sistema di gestione del file system nella console

Nella console di Instant Developer Cloud è stato implementato un nuovo sistema di navigazione del file system dei server, con una nuova veste grafica e tante nuove funzionalità. Il nuovo sistema di navigazione si attiva automaticamente quando viene usato su server IDE o di produzione aggiornati alla versione 22 di Instant Developer Cloud.

Le novità introdotte in questa nuova versione sono le seguenti.

- Albero di navigazione: sulla sinistra della schermata è stato aggiunto l'albero di navigazione del file system per il posizionamento rapido nelle cartelle.
- Gestione in schede: possibilità di lavorare contemporaneamente su più cartelle attraverso la visualizzazione a schede, disposte affiancate in tab.
- Pannello dettagli: è stata aggiunta la possibilità di attivare il pannello a destra dei dettagli del file o della cartella selezionata. I dati mostrati sono: nome, posizione, data ultima modifica, dimensione e l'anteprima.
- Filtri: è stata aggiunta la possibilità di filtrare nelle cartelle in lista per nome, data modifica e dimensione. I filtri possono essere applicati su tre livelli: filtrando file e cartelle nella cartella corrente, ma anche applicando la ricerca nelle eventuali sottocartelle presenti o in tutto il file system.
- Drag & drop: è stata aggiunta la possibilità di spostare file e cartelle trascinandole in un'altra cartella presente nell'albero di navigazione o tra le tab aperte.
- Menu contestuale: a seconda del punto su cui si clicca col tasto destro, si apre un menu contestuale appropriato alle operazioni disponibili. Ad esempio, cliccando su un file si ha la possibilità di copiarlo o rinominarlo, mentre cliccando in un punto vuoto della cartella è possibile crearne altre.
- Selezione multipla: è stata aggiunta la possibilità di effettuare la selezione multipla cliccando col mouse sui file e/o le cartelle, mantenendo premuto il tasto Cmd / Ctrl oppure Shift.



Ottimizzazione dell’accesso ai database offline nei dispositivi

Nota bene: questa modifica, resa necessaria da una breaking change di Chrome 99, può comportare un diverso comportamento delle applicazioni esistenti.

Contesto della modifica

Instant Developer Cloud mette a disposizione due modalità per utilizzare il database nelle applicazioni offline: attraverso il driver WebSQL o attraverso il driver nativo. WebSQL è un driver integrato all'interno dei browser, mentre il driver nativo è messo a disposizione direttamente dai sistemi operativi iOS e Android.

Fino alla versione 21.5, le applicazioni offline usano per default WebSQL su dispositivi Android e su browser, mentre usano il driver nativo su iOS. Questa differenza è dovuta al fatto che Apple ha rimosso WebSQL dalla sua WebView ormai da qualche anno.

Le nuove limitazioni introdotte in Google Chrome nella versione 99 non permettono più di usare il driver WebSQL quando il dispositivo è connesso all'IDE per il debug anche nel caso Android. Non hanno effetto sulle applicazioni installate nei launcher in produzione che quindi possono continuare a funzionare come prima.

Modifiche apportate all’accesso ai database

Nella versione 22, sono state quindi apportate le seguenti modifiche. Innanzitutto quando un dispositivo è connesso all’IDE per visualizzare l’anteprima delle applicazioni, utilizzerà sempre il driver nativo anche nel caso Android.

È stata inoltre modificata la proprietà *Driver offline* che ora presenta i seguenti valori.

- *WebSQL (se disponibile)*: è il valore di default per i progetti in versione precedente e non cambia il comportamento delle applicazioni Android quando installate, in quanto possono continuare ad utilizzare WebSQL. Si noti però che testando l’applicazione in anteprima collegata all’IDE, essa utilizzerà comunque il driver nativo.
- *Nativo (scelta consigliata)*: i dispositivi Android utilizzeranno sempre il driver nativo, sia in anteprima che in produzione. Questo valore è il default per i nuovi progetti.

Nota bene: impostando il valore *Nativo* per un progetto esistente, al momento dell’aggiornamento dell’applicazione in produzione, il database offline risulterà vuoto e dovrà essere nuovamente sincronizzato. Questo accade perché il driver WebSQL memorizza i propri database in un file system diverso da quello del driver Nativo.

Ottimizzazione della velocità di accesso ai dati

Nell'ambito di questa modifica è stata ottenuta un’importante accelerazione nell’accesso ai dati tramite driver nativo sia per sistemi iOS che Android. In alcuni casi si raggiungono **velocità fino a dieci volte superiori** alle versioni precedenti.

Generazione ottimizzata degli elementi visuali

Attraverso un'ottimizzazione del codice del framework di gestione del DOM remoto, è stata ottenuta un'importante accelerazione nella visualizzazione delle videate.

In particolare, nelle applicazioni online (sessioni web), **l'accelerazione è di circa 10 volte**, quindi una griglia complessa che nelle versioni precedenti richiedeva 500 ms per essere generata, ora impiega solo 50 ms. Questo migliora sensibilmente l'esperienza utente.

Nelle applicazioni locali (offline) **l'accelerazione è di circa 7 volte**. Questo permette di migliorare l'esperienza utente soprattutto su device Android, la cui webview è meno ottimizzata di quella iOS.

Revisione del framework di localizzazione linguistica

Il sistema di localizzazione è stato oggetto di una prima fase di miglioramenti che puntano a rendere la gestione della localizzazione più efficiente e precisa.

In seguito a questi miglioramenti è possibile che, aprendo la videata per la gestione delle traduzioni, alcune stringhe che prima erano *non traducibili* risultino *non definite*.

Per riportare il sistema allo stato precedente è sufficiente impostare queste stringhe come *non traducibili*.

Eliminazione di file ricevuti e non gestiti

Le applicazioni installate su server cloud memorizzano con un nome casuale i file ricevuti tramite POST multipart nella cartella *uploaded* del file system dell'applicazione. Dalla versione 22, se un'applicazione non implementa l'evento *onCommand*, i file ricevuti non vengono più memorizzati.

Se è stato implementato l'evento *app.onCommand* e non si vuole gestire i file ricevuti, è necessario inserire la seguente riga di codice:

```
App.Session.prototype.onCommand = function(request)
{
  request.files.forEach(f => f.remove());
}
```

Si segnala anche il seguente miglioramento: App: dimensione massima dei file ricevuti.

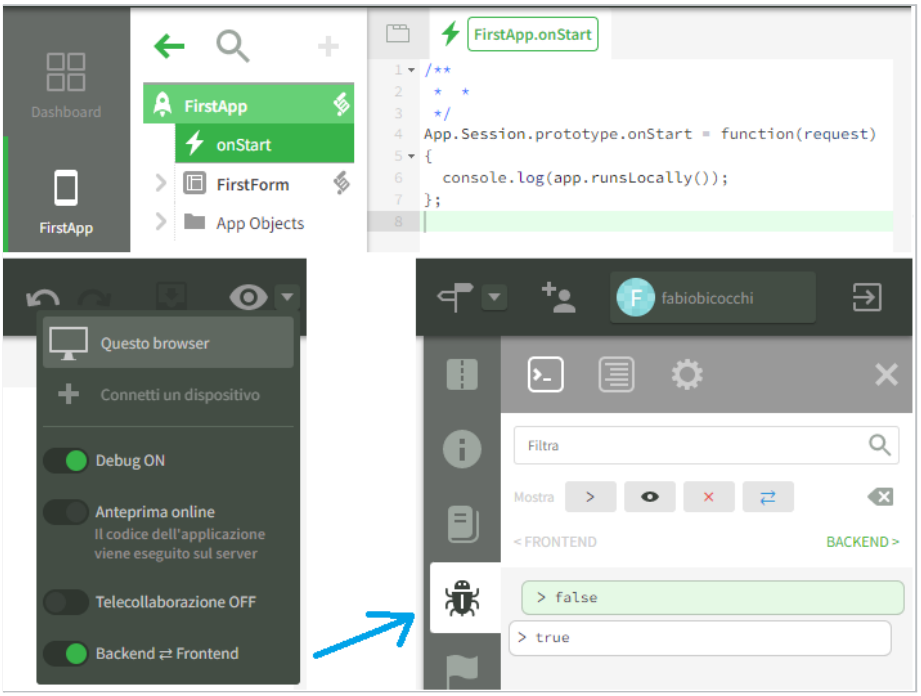
Per maggiori informazioni riguardo all'evento *onCommand* è possibile consultare il manuale 06-WebAPI.

Miglioramenti

Questa sezione contiene gli ulteriori miglioramenti apportati alla nuova versione di Instant Developer Cloud.

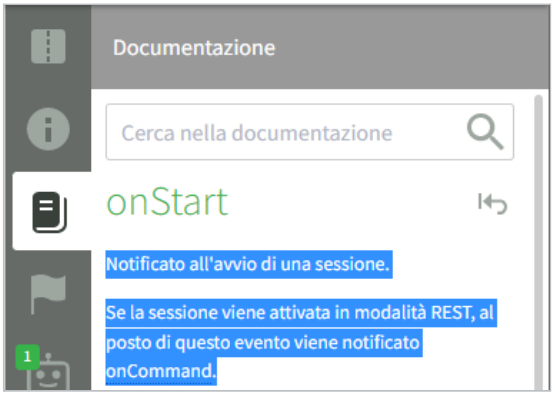
IDE: console di debug migliorata in modalità FE-BE

Per differenziare maggiormente i messaggi di log provenienti dal backend da quelli del frontend in anteprima FE-BE, è stata creata una nuova modalità di visualizzazione simile a quelle delle applicazioni di messaggistica.



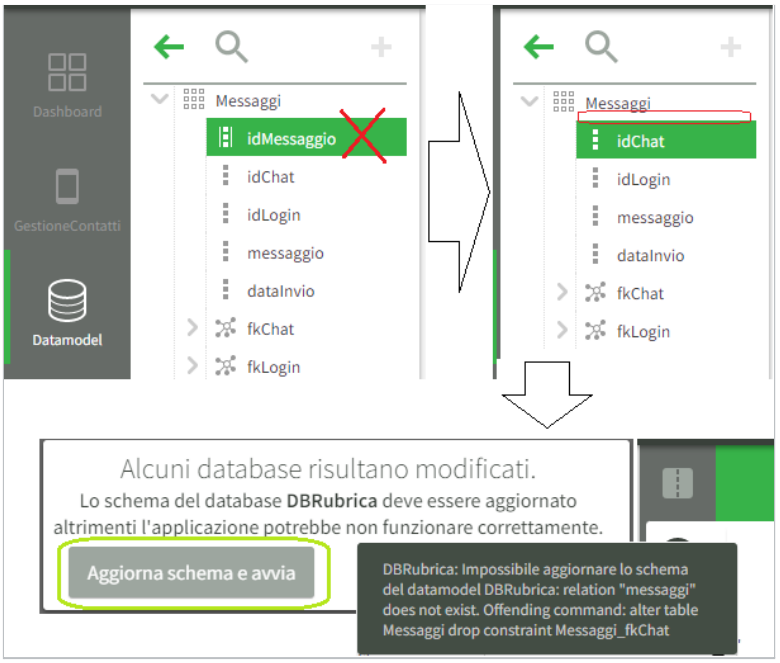
IDE: copiare il testo della documentazione

È stata abilitata la possibilità di selezionare il testo della documentazione ed eseguire il copia incolla solamente da tastiera, tramite Ctrl+C oppure Cmd+C.



DB: errori di aggiornamento schema più dettagliati

Aggiornando lo schema del database nell'ambiente IDE, in caso di errori di aggiornamento il comando che ha causato l'errore viene riportato nel messaggio di avvertimento.



Validazione delle foreign-key

Durante la validazione del progetto, che avviene prima dell'avvio dell'anteprima o della compilazione dell'applicazione, ora viene controllato che le foreign key contengano tutti i riferimenti ai campi primary key delle tabelle derivate, segnalando errore in caso diverso.

Nelle versioni precedenti, questa situazione non veniva controllata e si otteneva un errore in fase di aggiornamento dello schema.

Aggiornamento automatico delle foreign key

È stato aggiunto un sistema automatico per aggiornare le foreign key quando si modifica la chiave primaria della tabella da esse referenziata. In questo modo non è necessario modificare manualmente tali relazioni.

Aggiornamento schema in seguito a modifica di primary key

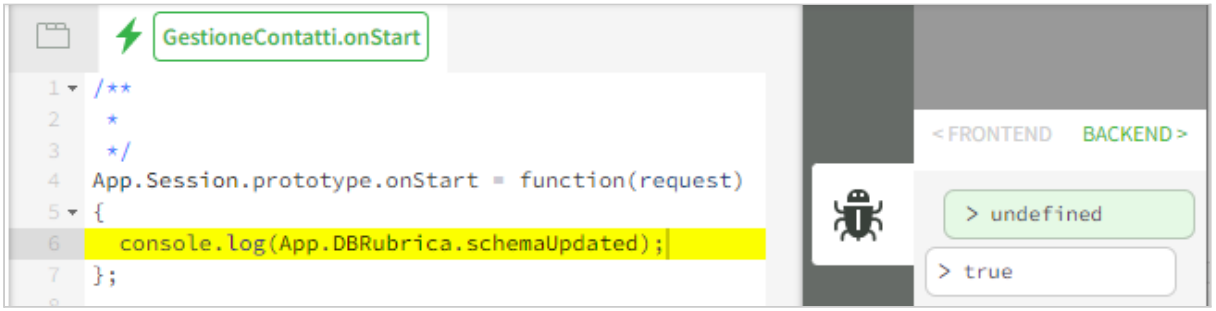
È stata corretta una situazione in cui nelle applicazioni offline l'aggiornamento dello schema falliva se si cancellava e ricreava un campo primary key con lo stesso nome del precedente, oppure se si aggiungeva un campo primary key ad una tabella esistente.

Controllo dell'aggiornamento dello schema di database offline

Alla classe *Database* è stata aggiunta la proprietà booleana *schemaUpdated* che indica se la struttura del database offline è stata aggiornata con successo.

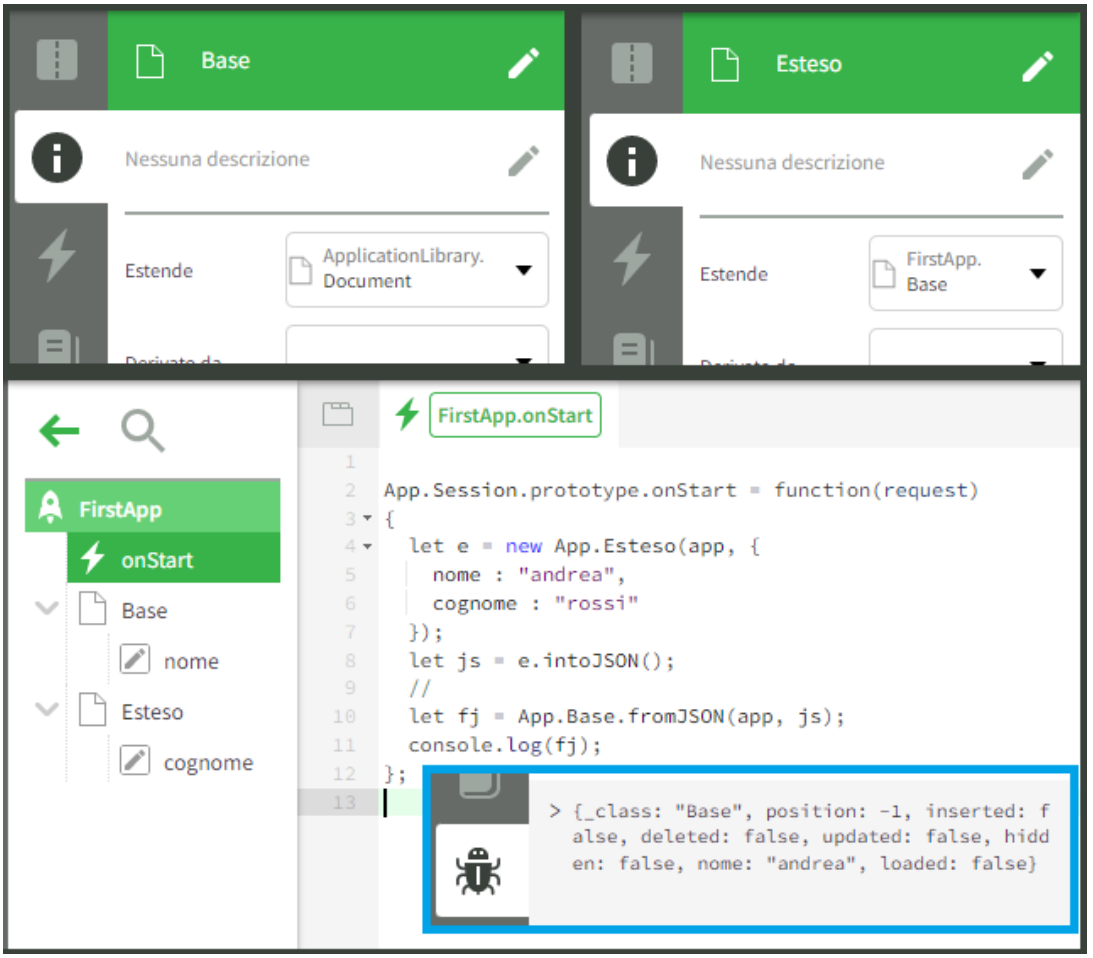
Questa proprietà può essere usata all'avvio dell'applicazione, solitamente nell'evento *app.onStart*, per sapere se la struttura del database è aggiornata.

Qualora l'aggiornamento non sia avvenuto con successo, è possibile utilizzare il metodo *App.Database.resetSchema* per risolvere la situazione. Si ricorda che dopo aver usato questo metodo, il database offline sarà vuoto e occorrerà sincronizzarlo completamente.



DO: caricare documenti da JSON

Il metodo *fromJSON* della classe *Document* è ora in grado di caricare istanze di documenti della classe base partendo dalla rappresentazione JSON di documenti di classi estese.



SYNC: configurazione dei dati inviati per tipo di documenti

Durante la sincronizzazione completa, il server sincronizza i documenti nel database locale inviando blocchi di record di grandezza pari al valore della proprietà *app.sync.maxMessages* impostata nella sessione locale (offline).

È ora possibile differenziare questo valore per tipo di documento. Per farlo è possibile personalizzare l'evento *onResyncClient* in due modi:

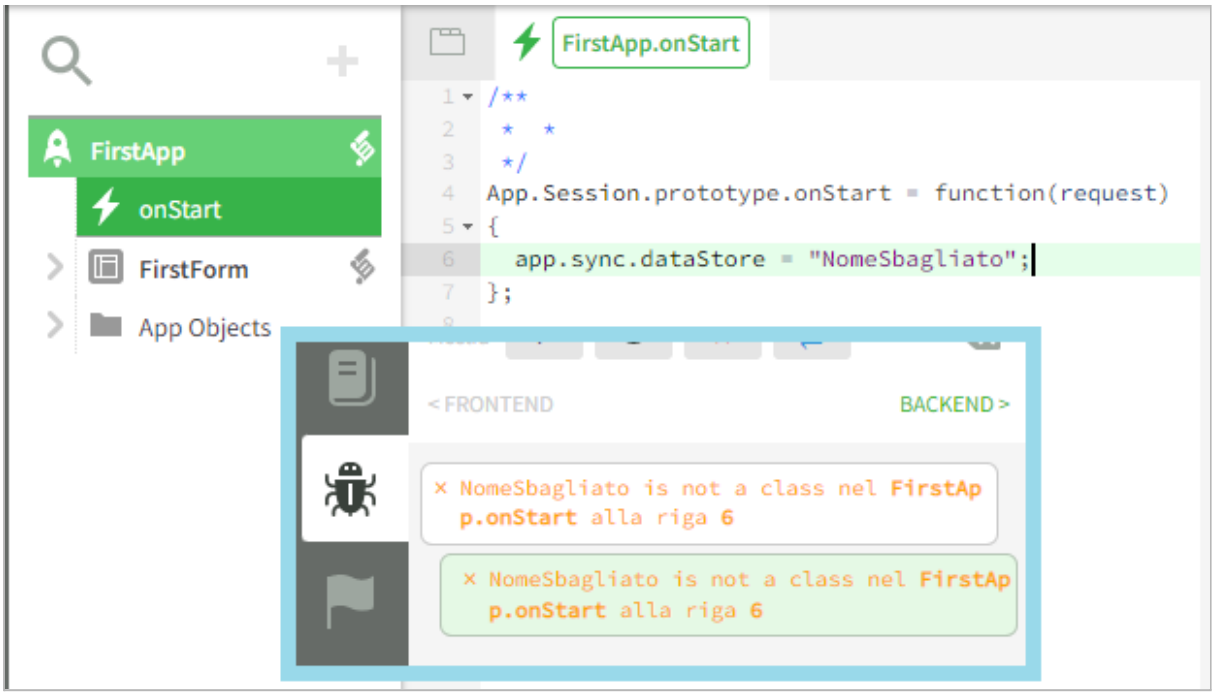
1) impostando il valore desiderato della proprietà *options.collections.maxRows*;
2) riempiendo la proprietà *options.collection* con la quantità di dati desiderata.

Si ricorda che se si utilizza l'evento *onResyncClient* per parzializzare i dati inviati, è necessario anche notificare al framework quando l'invio dei dati è completato impostando il parametro *options.partial* a *false*.

SYNC: controllo della proprietà dataStore

La configurazione del framework di sincronizzazione prevede di impostare la proprietà *app.sync.dataStore* al nome della classe *Database* che conterrà i dati relativi ai messaggi di sincronizzazione.

L'impostazione di questa proprietà ad un nome di classe non presente nel progetto è stata ora protetta, mentre nelle versioni precedenti generava un'eccezione JavaScript. In ogni caso la sincronizzazione non può funzionare senza l'impostazione corretta.



Per maggiori informazioni riguardo la proprietà *dataStore* è possibile consultare il manuale 07-Sincronizzazione.

SYNC: aggiornamento automatico di proprietà derivate

I documenti sono in grado di aggiornare automaticamente i valori delle proprietà derivate quando cambiano le proprietà che ne determinano il valore. Questo si verifica solo se il cambiamento avviene sul documento stesso. Ad esempio, un documento Prodotto può contenere il nome della Categoria a cui esso appartiene come proprietà derivata: quando l'id della categoria del Prodotto cambia, il nome della Categoria viene automaticamente aggiornato.

Questo aggiornamento automatico non avviene invece se cambia il documento puntato, cioè se viene variato il nome della Categoria, i documenti Prodotto già caricati non aggiornano automaticamente il nome della Categoria che essi contengono come proprietà derivata.

Per migliorare questo comportamento, nella versione 22 i documenti reagiscono automaticamente all'evento *onDocUpdate*, aggiornando le proprie proprietà derivate quando rilevano che il documento per cui viene notificato *onDocUpdate* è a loro correlato.

È quindi possibile ottenere l'aggiornamento automatico delle proprietà derivate anche quando cambia il documento puntato. Per ottenere questo comportamento è necessario notificare la modifica chiamando il metodo *notifyDocUpdate* del documento nell'evento *onSave* in fase *aftersave*, come mostrato nell'esempio seguente:

```
App.DBItaliaLibreria.Comuni.prototype.onSave = function(options)
{
  if (options.phase===App.Document.savePhases.afterSave && this.updated) {
    this.notifyDocUpdate();
  }
};
```

Per maggiori informazioni riguardanti l'evento *onDocUpdate* è possibile consultare il manuale 07-Sincronizzazione.

IonicUI: Miglioramenti IonFooter per iPhone

È stata aggiunta la classe CSS *footer-padding* che su iOS permette di estendere l'altezza di un oggetto contenuto nel footer in modo da coprire l'area di schermo non utilizzabile nel lato inferiore.

Tale classe va applicata ai figli di primo livello di *IonFooter* ed è necessario usarla solo se l'applicazione viene utilizzata a schermo intero chiamando il seguente metodo:

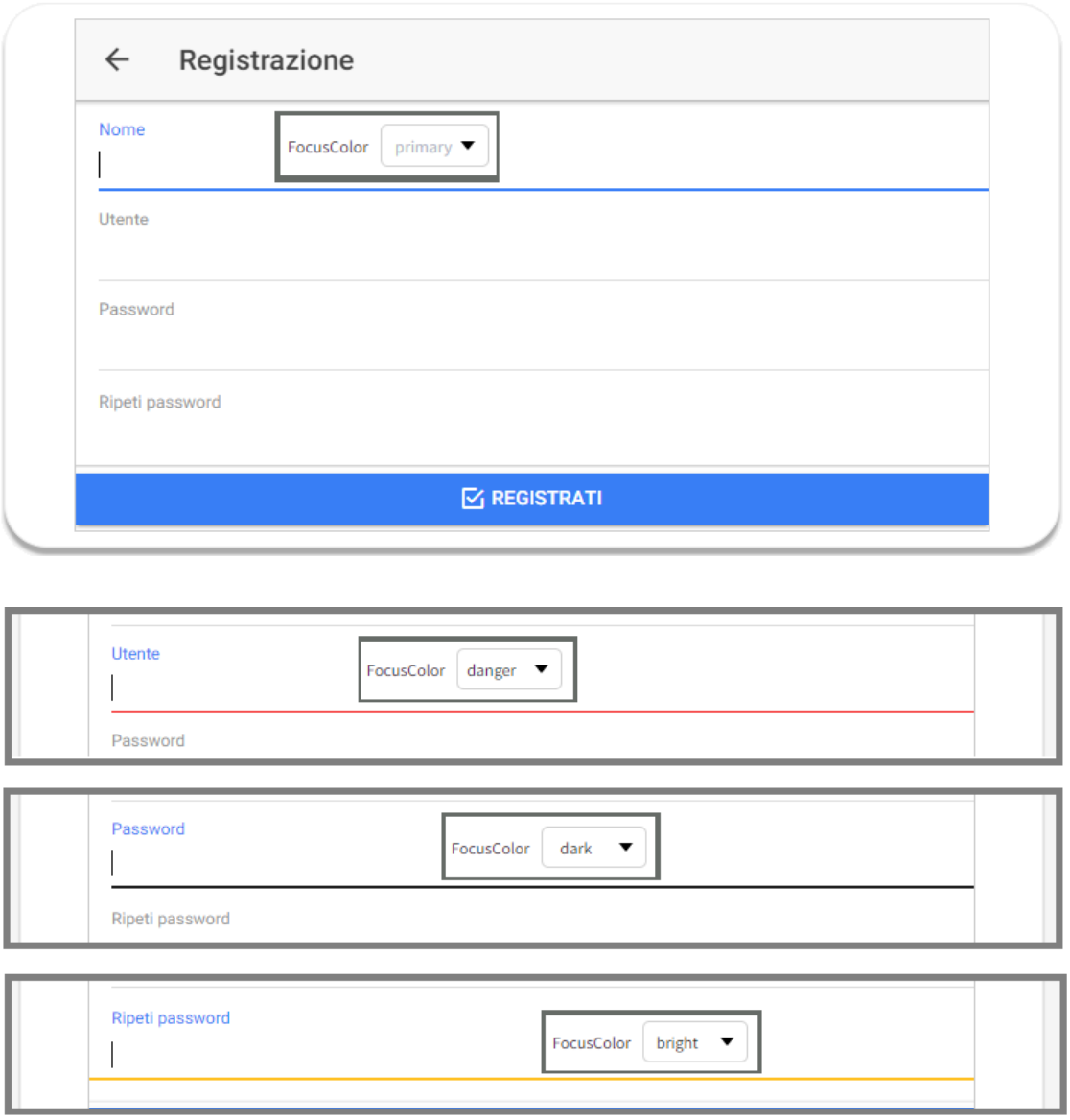
```
app.device.statusBar.overlaysWebView(true);
```

Si consiglia di attivare lo schermo intero ed utilizzare la classe *footer-padding* per rendere l'applicazione più simile a quelle native iOS.

IonicUI: proprietà focusColor

Agli elementi *IonInput, IonCheck, IonToggle, IonRadio, IonSelect, IonAutoComplete* e *IonRange* è stata aggiunta la proprietà *focusColor* per permettere di personalizzare per ogni elemento il colore di sottolineatura del tema grafico *MaterialDesign*.

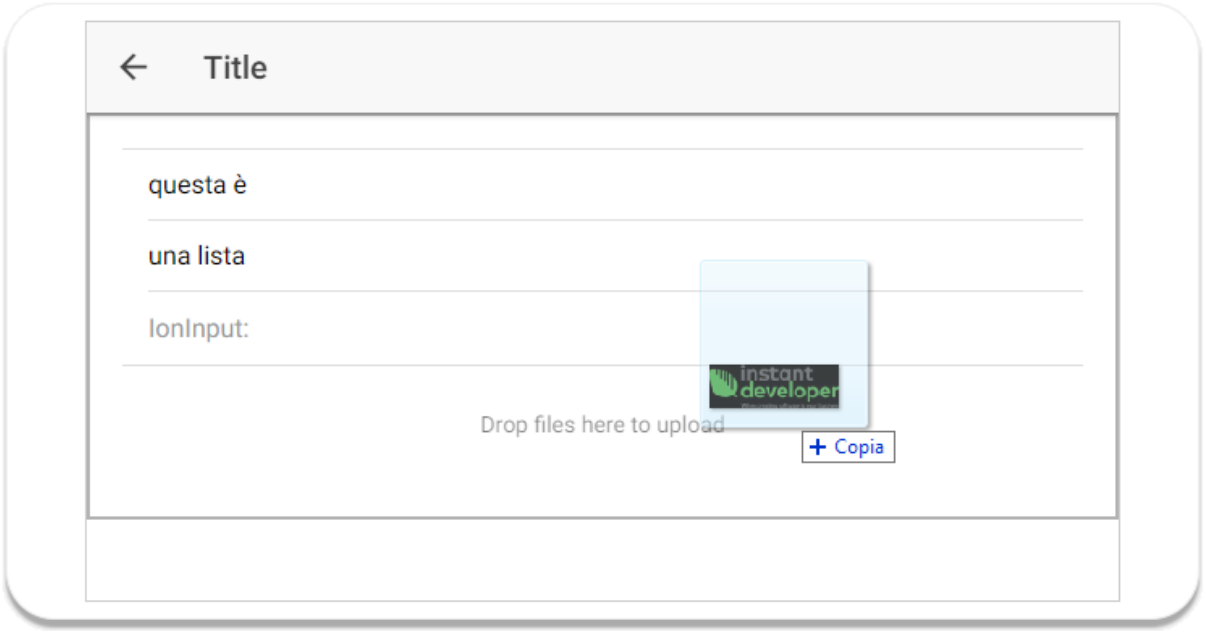
In precedenza era possibile configurare il colore solo a livello di tema, cioè per tutti gli elementi.

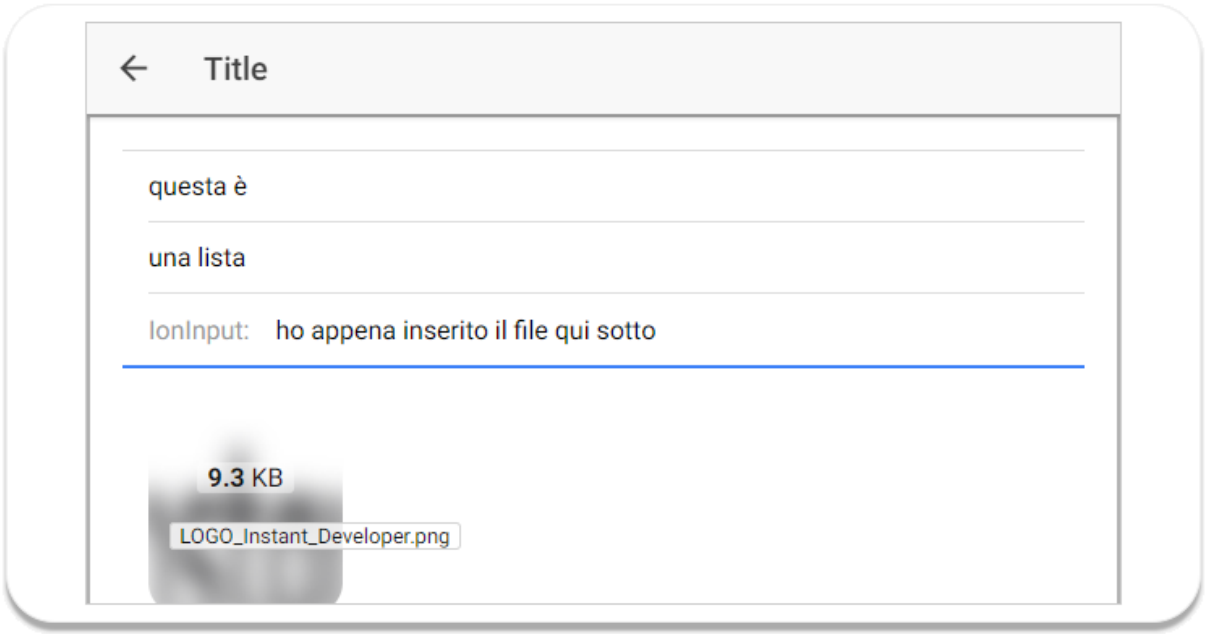


App: DropZone come container di elementi

L'elemento *DropZone* è ora utilizzabile come container di altri elementi. In questo modo è possibile rendere sensibili al trascinamento intere parti della propria interfaccia utente, invece che dover definire una zona sensibile a parte.

È quindi possibile realizzare meccanismi di trascinamento in stile *Google Drive*.





App: miglioramenti HtmlEditor

Localizzazione

All'elemento *HtmlEditor* è stata aggiunta la proprietà *lang* che permette di scegliere la lingua in cui visualizzare l'editor: in particolare verranno localizzati i tooltip dei comandi nella lingua impostata. Se la proprietà non viene valorizzata, il sistema di default utilizza la lingua del browser.
Può essere configurata con valori nel seguente formato: *it*, *en*, *fr*, *es*, *de* e così via
Al link seguente è presente la documentazione con la lista di tutte le lingue disponibili:
https://github.com/Alex-D/Trumbowyg/tree/develop/src/langs .

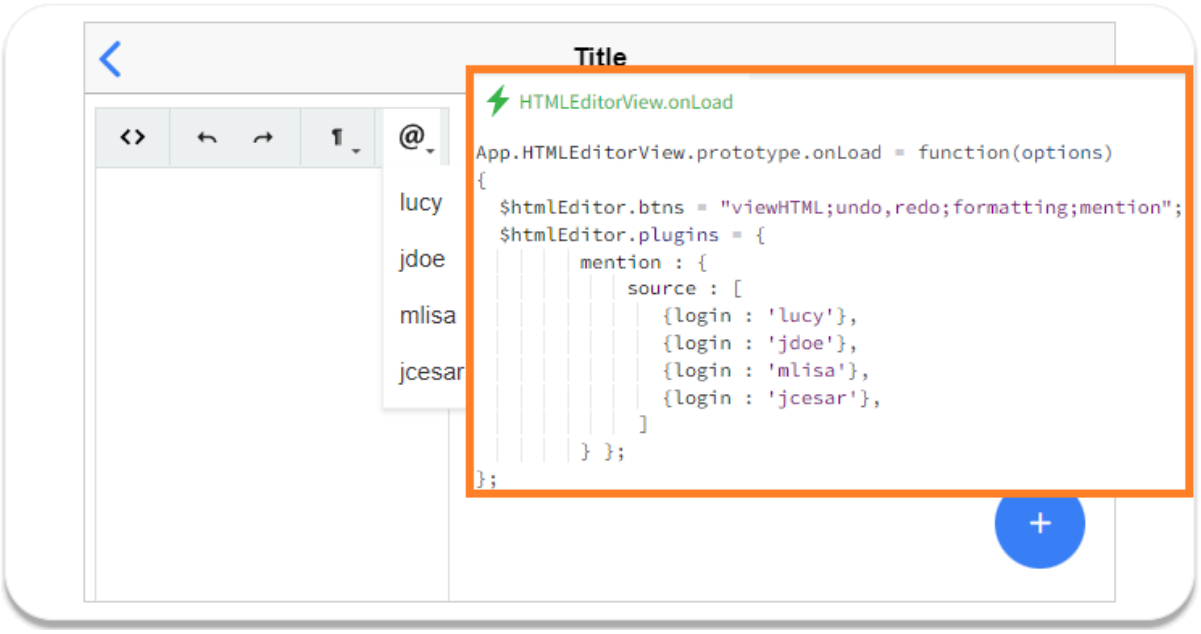
Nuovi plugin

La proprietà *plugins* permette di configurare i plugin attivi per un elemento *HtmlEditor*.
Questa proprietà è di tipo JSON e deve essere valorizzata nel codice dell'applicazione.

I plugin configurabili sono i seguenti.

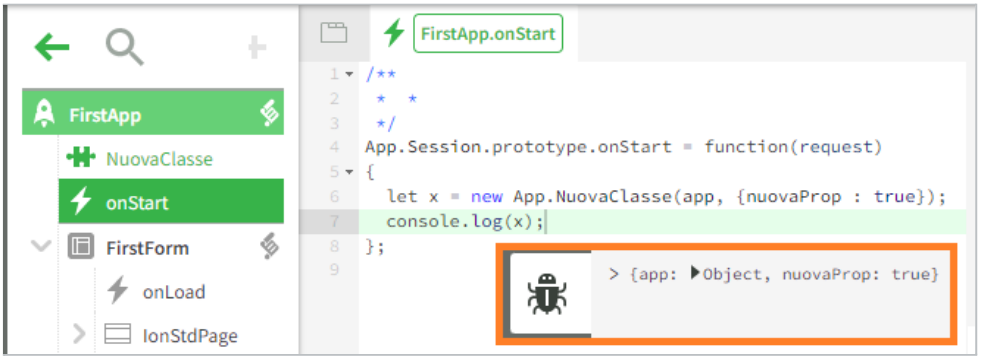
- colors: permette di scegliere il colore del testo.
- fontfamily: permette di scegliere il tipo di font da usare.
- fontsize: permette di scegliere la dimensione del carattere da usare.
- history: funzionalità avanzate di undo/redo.
- mention: permette di usare un sistema di menzioni.
- indent: permette di gestire l'indentazione del testo.
- mathml: permette di inserire formule matematiche.

Per poter utilizzare i plugin configurati è necessario includere i relativi pulsanti nella toolbar dell'elemento *HtmlEditor* impostando correttamente la proprietà *btns*. Per ulteriori informazioni è possibile consultare la documentazione ufficiale al link:
https://alex-d.github.io/Trumbowyg/documentation/plugins/ .



App: proprietà nel costruttore delle classi

Ai costruttori delle classi è stata aggiunta la possibilità di passare come secondo parametro un oggetto contenente le proprietà da impostare sull'istanza in fase di creazione. Questa modifica riguarda solo le classi generiche e non i documenti.



App: dimensione massima dei file ricevuti

È stata impostata una soglia di default di 50 MB alla dimensione massima dei file ricevuti dall'applicazione tramite richieste di tipo multipart.

Per modificare questa soglia è possibile impostare il parametro di runtime *maxUploadFileSize* indicando la massima dimensione ammessa in byte.

Il parametro può essere definito a livello di server, per imporre la soglia su ogni file, oppure a livello di applicazione, se è necessario fissare un limite specifico ad hoc.

Per maggiori informazioni riguardo a come impostare i parametri di runtime è possibile consultare il manuale 17-I server di produzione.

App: miglioramento a sendResponse

Il metodo *sendResponse* ora emette automaticamente l'header *content-Type* in funzione del contenuto della risposta.

- *contentType* = *text/plain*, se la risposta è di tipo *string*.
- *contentType* = *application/json*, se la risposta è di tipo *object*.

Per maggiori informazioni riguardanti le risposte alle chiamate REST è possibile consultare il manuale 06-WebAPI.

App: sessionTimeout della server session predefinita

La proprietà *sessionTimeout* della server session predefinita non ha più effetto. Per maggiori informazioni riguardo le server session è possibile consultare il manuale 17-I server di produzione.

App: determinare lo spazio libero del disco dati

Per conoscere lo spazio libero del disco dati è ora possibile utilizzare la funzione *App.Utils.getFreeDiskSize* che restituisce la quantità di byte disponibili sul disco dove è ospitata l'applicazione. Questo metodo è disponibile solo nelle applicazioni online; eseguendolo offline causa errore.

App: encoding di default

L'encoding viene inizializzato a *utf-8* quando si chiamano i metodi *readAll*, *readLine* e *write* con stringa.
Precedentemente questi metodi generavano errori se venivano richiamati in applicazioni online in cui non era stato specificato l'encoding.

BackgroundLocation: aggiunta funzione checkStatus

Al plugin *backgroundLocation* è stata aggiunta la funzione *checkStatus* che restituisce informazioni sullo stato del servizio di background location.

Il motivo per cui è stato introdotto è che nelle versioni precedenti era possibile sapere se il GPS era attivo sul dispositivo (tramite il metodo *isLocationEnabled* del plugin), ma non se l'applicazione lo stava usando per rilevare la posizione in quel momento (il GPS può essere attivo, ma questo non significa che qualche applicazione lo stia usando).
La funzione *checkStatus* restituisce un oggetto che permette di conoscere tutte queste informazioni, a seconda del dispositivo da cui viene eseguita l'applicazione:

- *isRunning*: valore booleano che indica se è attualmente in corso il rilevamento della posizione (da browser e dispositivi mobile);
- *locationServicesEnables*: valore booleano che indica se i servizi di rilevamento della posizione sono abilitati (solo da dispositivi mobile);
- *authorization*: valore numerico che indica lo stato di autorizzazione del servizio (solo da dispositivi mobile).

Si ricorda che è necessario usare un launcher aggiornato alla versione 22.0. Per maggiori informazioni sui possibili stati di autorizzazione è possibile consultare la documentazione ufficiale al link:
https://github.com/mauron85/cordova-plugin-background-geolocation#readme .

Per maggiori informazioni riguardo al plugin *backgroundLocation* è possibile consultare il manuale 15-Launcher.

Correzioni della versione 22.0

IDE

Code Editor: corretto errore generato da riga con commento finale

Nel Code Editor è stato intercettato un particolare caso in cui l'applicazione falliva all'avvio. Il malfunzionamento si manifestava se, all'interno di un metodo, si scriveva un'espressione del tipo:

```
return yield app.[asynch method]([params list]) // comment
```

senza il carattere *; (punto e virgola)* dopo la parentesi chiusa.
Malfunzionamento relativo alla richiesta di assistenza 001851-2021.

Corretto caricamento risorsa tramite drag & drop

È stata effettuata la correzione sul caricamento dei file come risorsa tramite drag & drop: in alcuni casi veniva mostrato nella videata di dettaglio della risorsa il messaggio di errore *Can't get file info (file is missing?)*.

Corretta formattazione dei commenti

In un caso molto particolare, ad ogni modifica di un metodo, una riga commentata veniva indentata di alcuni spazi. Il malfunzionamento si manifestava se era stato scritto codice con Promise e arrow function.
Malfunzionamento relativo alla richiesta di assistenza 002006-2021.

Corretta importazione dei componenti

È stata effettuata una correzione sull'importazione dei componenti che contengono nel nome il carattere **_** *(underscore)*.
Precedentemente l'importazione falliva: una volta effettuata l'importazione, non si riusciva ad eseguire l'importazione e veniva segnalato l'errore *L'importazione del componente non è riuscita*.

Corretta gestione delle risorse nella copia tra progetti

Quando si effettua la copia di oggetti tra progetti differenti vengono ora gestite correttamente anche le risorse.
Precedentemente, se le risorse erano file su disco salvate sul server, queste non venivano copiate.

Correzione al debug

Definendo una funzione locale tramite operatore => non vengono più generati errori di compilazione attivando il modulo di debug. Precedentemente il sistema non gestiva le variabili definite dentro la callback.
Malfunzionamento relativo alla richiesta di assistenza 002100-2021.

Corretta definizione di oggetti che si riferiscono a classi duplicate

Se si duplicava una classe e questa era stata utilizzata come tipo di dato in una dichiarazione di variabile, questa veniva cambiata puntando alla nuova classe duplicata. Per esempio, se si duplicava la classe *Categories* e nel codice era stata definita una variabile:

```
var v = this.instance; // type:Categories
```

la dichiarazione veniva cambiata automaticamente in

```
var v = this.instance; // type:Categories1
```

dove *Categories1* era la nuova classe ottenuta dalla duplicazione di *Categories*.
Malfunzionamento relativo alla richiesta di assistenza 002743-2021.

Correzione sui componenti in un progetto forkato

Se si importava o si aggiornava un componente che conteneva una risorsa in un progetto forkato, si potevano avere problemi se si inviava una pull request al progetto master.
In alcuni casi il file della risorsa non veniva inviato al master.

Corretta importazione dei componenti con risorse linguistiche

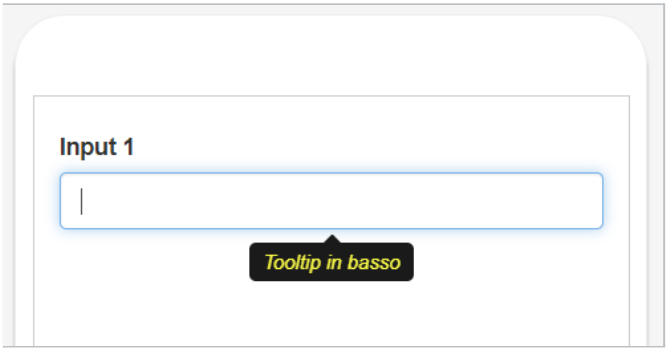
Importando un componente vengono caricate correttamente a run-time anche le risorse linguistiche.
Precedentemente, se all'interno delle proprie videate non erano stati utilizzati oggetti visuali contenuti nel componente, le risorse linguistiche non venivano caricate e di conseguenza le stringhe presenti nel componente non erano tradotte a run-time.

Bootstrap

Corretto utilizzo HTML in tooltip di BSInput

Se si impostava il tooltip di un elemento *BSInput* utilizzando codice HTML, esso veniva mostrato correttamente nell'IDE ma non a run-time. Ora la visualizzazione è la stessa sia nell'IDE che a run-time. Ricordiamo che il motore dei tooltip di Bootstrap richiede l'uso di un JSON con *html:true* per gestire correttamente l'inclusione di codice HTML, come mostrato di seguito.

```
{"placement" : "bottom", "title" : "<font color='yellow'><em>Tooltip in basso</em>", "html" : true}
```



Cloud Connector

Corretto metodo changeCloudConnectorConfig

Nel Cloud Connector, se si usava il metodo *changeCloudConnectorConfig* e la password di un database era cambiata, tale modifica non veniva applicata.

Device

Corretta sovrascrittura della tastiera all’app

Se si utilizzava il metodo *app.device.statusBar.overlaysWebView(true)* per visualizzare l'applicazione a tutto schermo, su dispositivi Android la tastiera veniva aperta sovrapposta all'applicazione, coprendola parzialmente. Si ricorda che dalla versione 21.5 è possibile e consigliabile utilizzare applicazioni a tutto schermo sia per dispositivi iOS che Android.

Impostazione tempo blur chiusura tastiera Android

È stato aggiunto il parametro di tema *keyblurtime* per impostare il tempo tra la chiusura della tastiera e l’azione di *blur* che avviene sull'elemento di input per dispositivi Android. Il valore predefinito è 0. Normalmente non è necessario utilizzare questo parametro. In alcuni dispositivi, con particolari tastiere non standard, se si utilizza il sistema di dettatura vocale, le caratteristiche della tastiera interferiscono con il funzionamento del framework causando la chiusura e la riapertura della stessa. In questi casi si consiglia di impostare a 50 o 100 il valore di questo parametro.

Document Orientation

Corretto metodo intoJSON con blob e oggetti

Il metodo *intoJSON* ora serializza correttamente i valori originali in caso di blob e oggetto. Precedentemente non salvava in modo uguale i valori attuali rispetto a quelli originali.

Corretto salvataggio remoto sulle collection

In seguito alla correzione, se viene effettuato un salvataggio remoto di una collection in cui lato server è stata valorizzata la proprietà *childLevel*, questa viene gestita correttamente. Precedentemente non ne veniva considerato il valore assegnato e lo impostava comunque 9999.

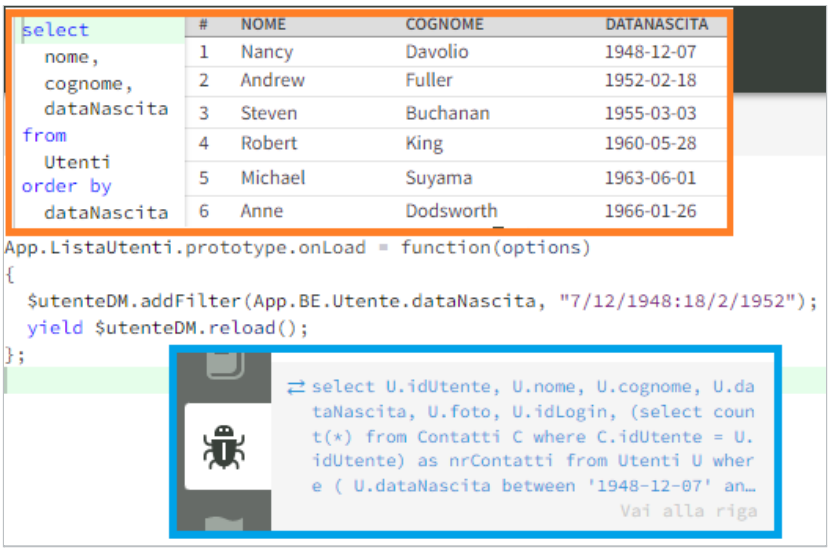
DataMap

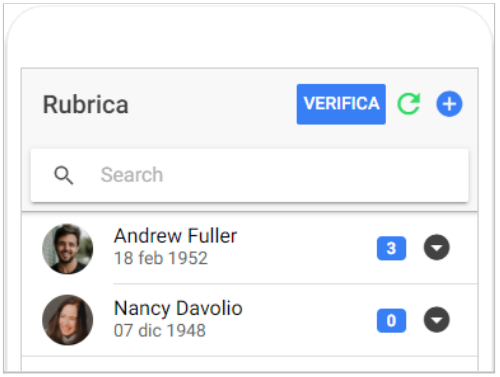
Corretto caricamento datamap innestate DO

È stato protetto un caso in cui si ha una datamap DO di testata che contiene una datamap innestata, ma svincolata da quella che la contiene. Precedentemente, se la datamap innestata non conteneva filtri relativi a quella di testata si otteneva un errore JavaScript.

Corretto filtro per intervallo date

Nelle datamap è stato corretto il filtro QBE per intervalli di date, come mostrato nell'esempio seguente: *7/12/1948:18/2/1952*. Si ricorda che, dopo aver applicato il filtro, è necessario chiamare il metodo *reload* sulla datamap. Precedentemente poteva essere generata una eccezione JavaScript.





Ionic

Corretto placeholder dell'elemento IonSelect

È stata corretta una regressione per cui la proprietà *placeHolder* dell'oggetto *IonSelect* non veniva visualizzata anche se impostata.

Corretta gestione pulsante clear dell'elemento IonSearchBar

Quando si fa clic sul pulsante *clear* dell'elemento *IonSearchBar* ora viene notificato immediatamente l'evento *onChange* con il parametro *event.clear* impostato a *true*. Precedentemente l'evento non veniva notificato subito, ma solo dopo che l'elemento aveva perso il fuoco.

Corretta visualizzazione elemento IonSelect in un caso particolare

È stato corretto il funzionamento dell'elemento *IonSelect* nel caso in cui la lista include anche il valore 0 come numero intero. Se non ci sono valori selezionati, l'elemento ora non mostra nulla, mentre precedentemente selezionava il valore 0.

Corretto inserimento dello 0 da tastierino numerico

È stato corretto l'inserimento dal tastierino numerico del numero 0 come parte intera di un numero decimale. Precedentemente lo zero davanti alla virgola veniva rimosso. Ad esempio è ora possibile visualizzare *0,55* mentre in precedenza veniva mostrato *,55*.

Corretta gestione proprietà textColor della classe ProgressBar

È stata corretta la gestione della proprietà *textColor* della classe *ProgressBar* che ora funziona correttamente anche quando vengono impostate le proprietà *fromColor*, *toColor* e *textValue*.

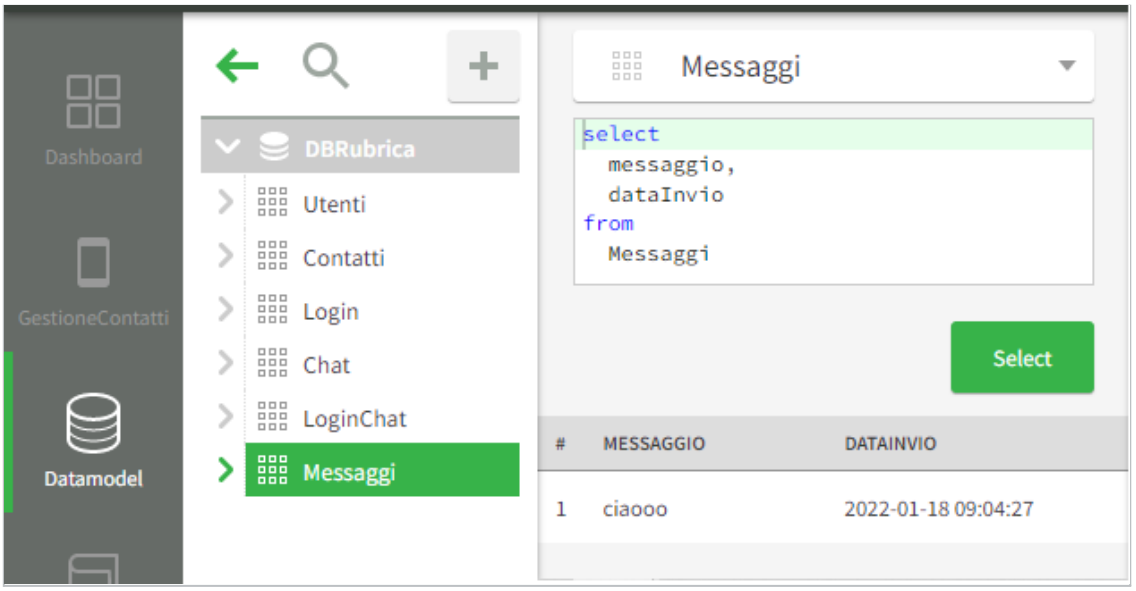
SQLView

Corretta importazione CSV

È stata migliorata l'importazione dei file CSV eseguita dall'SQLView quando i dati da importare sono un numero considerevole.

Corretta visualizzazione dati in formato datetime

È stata modificata la formattazione dei campi di tipo *datetime* in *YYYY-MM-DD hh:mm:ss*. Precedentemente veniva mostrato il formato ISO *YYYY-MM-DDThh:mm:ss.sssZ*.



Applicazione

Placeholder dell’elemento autocomplete

Il *placeholder* dell'elemento *autocomplete* è ora traducibile. Nota bene: non riguarda gli elementi *IonAutoComplete*.

Impostazione proprietà tema allowCrossDomainCommands

La proprietà di tema *allowCrossDomainCommands* è stata resa di tipo boolean: è ora possibile impostarla sia con il valore *true* che *“true”*. Attivando questa proprietà, se una sessione browser è contenuta in un *iframe* viene consentito l’invio di eventi e comandi a partire dal frame contenitore.

Corretta gestione metodo app.open passando URL con spazi

È ora possibile passare URL contenenti spazi al metodo *app.open* . Gli spazi verranno sostituiti in automatico con il carattere + *(più)*.

Corretto l'evento onChange della classe Device su browser

Ora l’evento *device.onChange* viene notificato anche su browser quando cambia la proprietà *device.networkState*. Precedentemente questo avveniva solo usando l’applicazione in un dispositivo.

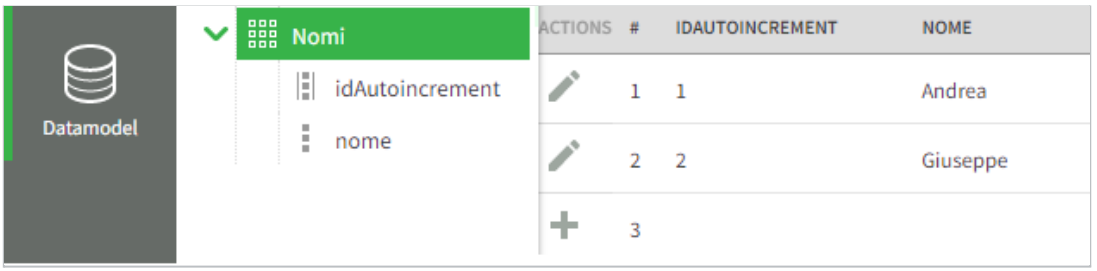
Aggiunta protezione all’avvio del server

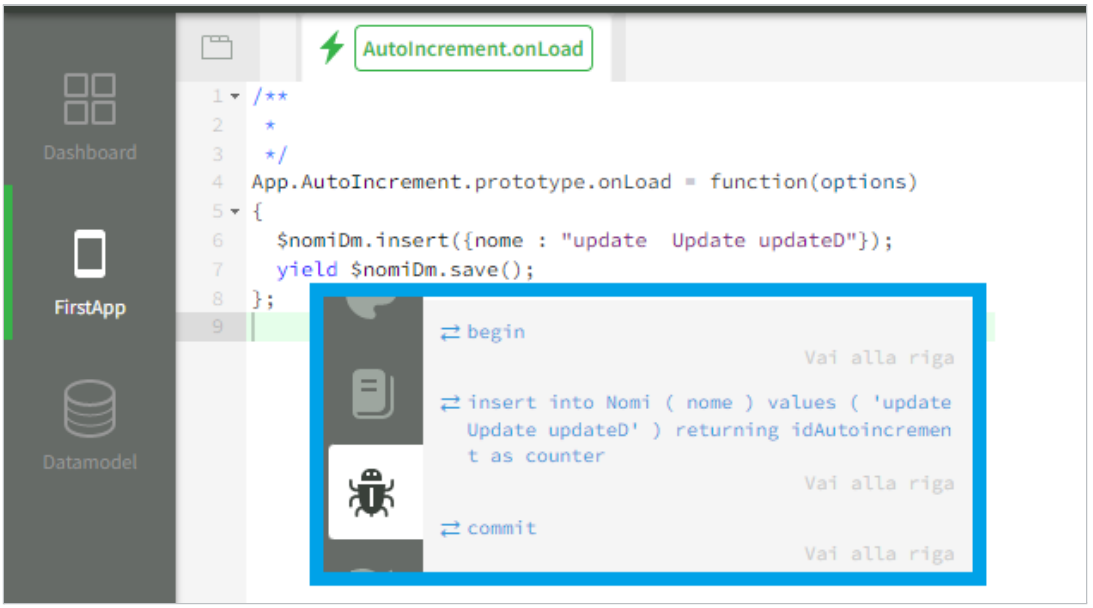
È stato protetto l’avvio di un server in un caso particolare di file di configurazione vuoto. Malfunzionamento relativo alla richiesta di assistenza 002296-2021.

Protetto l’inserimento di stringhe contenenti la parola update

È stata corretta l’esecuzione di un comando di inserimento SQL quando i valori da inserire contenevano la stringa *update.* Il problema si verificava durante il salvataggio di un documento, se la tabella aveva una primary key di tipo contatore, nei database di tipo SQLServer.

In particolare ora viene letto il valore della chiave attribuita dal database al numero del record e quindi il documento in memoria contiene tale valore nella proprietà corrispondente. In precedenza la proprietà rimaneva non valorizzata.




Dashboard
FirstApp
Datamodel
AutoIncrement.onLoad
/**
*
*/
App.AutoIncrement.prototype.onLoad = function(options)
{
$nomiDm.insert({nome : "update Update updateD"});
yield $nomiDm.save();
};
begin
Vai alla riga
insert into Nomi (nome) values ('update Update updateD') returning idAutoincrement as counter
Vai alla riga
commit
Vai alla riga


Datamodel
Nomi
idAutoincrement
nome
ACTIONS
#
IDAUTOINCREMENT
NOME
1
1
Andrea
2
2
Giuseppe
3
3
update Update updateD
4