Versione 19.5 - Note di rilascio

Nuova offerta commerciale

A partire dal 1 luglio 2019 entra in vigore il nuovo listino per Instant Developer Cloud che verrà applicato ai nuovi acquisti. Tutti gli acquisti effettuati prima di questa data manterranno le condizioni precedenti finché rimangono attivi.

Le modifiche principali sono le seguenti:

- L'installazione di applicazioni sviluppate nei server IDE di community è possibile solo sui sui server di taglia XS. I progetti creati su tali server IDE avranno un limite di 20 videate per progetto e non potranno essere trasferiti sui server di organizzazione.
- Il server di taglia XS potrà essere utilizzato solo per scopi di demo e test di applicazioni, infatti avrà un limite di sessioni contemporanee pari a 3.
- I server IDE di organizzazione acquistati nell'anno 2019 avranno un limite di sessioni di editing contemporanee pari al numero di licenze IDE acquistate dall'organizzazione.

Aggiornamento sistemi operativi

La versione 19.5 è l'ultima versione di Instant Developer Cloud in grado di supportare Node versione 6. A partire dalla prossima versione verrà supportato solo Node 10 in quanto il supporto ufficiale per Node 6 è terminato a maggio 2019.

Per i server gestiti nel cloud di Instant Developer (sia IDE che PROD) è quindi necessario aggiornare il sistema operativo. La procedura è la seguente:

1) Aggiornare i server IDE alla versione 19.5 e testare le proprie applicazioni.
2) Aggiornare i server PROD alla versione 19.5 ed installare le proprie applicazioni.
3) Per ogni server IDE o PROD inserire una richiesta di assistenza con pre-valutazione indicando il nome del server da aggiornare. Tutte queste richieste saranno gestite gratuitamente fino al 31 dicembre 2019 per tutti i server di taglia S o superiore. Durante l'intervento di assistenza, il server rimarrà spento.

I nuovi sistemi operativi sono basati su tecnologia Docker di Google. Dalla versione 20 Instant Developer Cloud avrà la possibilità di aggiornare i sistemi operativi in maniera quasi trasparente limitando il tempo di indisponibilità del server a qualche minuto.

Team Works

La versione 19.5 contiene un nuovo sistema di Team Working, che, lavorando in maniera più simile allo standard GIT, consente operazioni più veloci e sicure, soprattutto in presenza di branch multipli.

L’aggiornamento dei server IDE alla versione 19.5 richiede alcune accortezze a causa delle modifiche al funzionamento del Team Works.

In particolare, prima di aggiornare un server IDE alla versione 19.5 è necessario che:

1) Tutti i progetti del server non abbiano fork. Se ci sono dei fork è quindi necessario consolidare le versioni e cancellare i fork.
2) Tutti i progetti del server non abbiano alcun branch a parte il master.

Dopo aver aggiornato il server, è possibile effettuare nuovamente i fork dei progetti.

Novità della versione 19.5

IDE

❝ È stato aggiunto il tema scuro. Nelle impostazioni si può selezionare il tipo di tema da usare tra *Automatico, Chiaro e Scuro*. La gestione automatica utilizza il tema configurato nel browser o nel sistema operativo.

❝ È stato migliorato l’algoritmo utilizzato dal modulo Team Works per la creazione dei commit per renderlo più simile a quello utilizzato da Git.

APP

❝ È stato aggiunto un nuovo elemento visuale *AceEditor* per l'editing di codice o testi complessi, anche molto lunghi. È richiesto l'aggiornamento di *coreLibraries* per ottenere la definizione.

❝ È stato aggiunto un nuovo elemento visuale *Mermaid* per mostrare vari tipi di diagrammi a partire da definizioni testuali. È richiesto l'aggiornamento di *coreLibraries* per ottenere la definizione.

❝ È stata aggiunta la funzione *app.device.checkForUpdates* che permette di lanciare la verifica della presenza di aggiornamenti dell'applicazione. Deve essere utilizzata per applicazioni distribuite tramite launcher per il quale è attiva la configurazione automatica degli aggiornamenti. Solitamente la ricerca viene effettuata quando l'applicazione parte oppure quando essa torna in foreground.

SYNC

❝ È ora possibile utilizzare chiamate remote fra documenti di due applicazioni entrambe installate su server cloud.

❝ La sincronizzazione ora gestisce il caso di applicazioni multi-tenant sia su singolo database che su database separati. Qualora sia necessario tenere separati i dati di

sincronizzazione è sufficiente impostare la proprietà *sync.tenant* nell'evento *sync.onConnect*.

❝ È stato aggiunto l'evento *onMissingDocument* per la gestione di documenti mancanti durante la sincronizzazione. Questo può avvenire se un documento cambia "dominio di sincronizzazione" e, quando questo avviene, il problema può essere gestito tramite l'evento suddetto. Il framework offre anche una gestione di default che consiste nel recuperare il documento tramite chiamata remota al server nel cloud. Perché questo avvenga occorre aver attivato le chiamate remote sul documento in fase di sincronizzazione.

❝ È ora possibile rendere più veloce la sincronizzazione differenziale all'avvio della sessione attivando la proprietà *sync.DO.fastDiff*. Questa modalità opera direttamente sul database senza passare dai documenti; è un metodo è decisamente più veloce, ma non è in grado di aggiornare automaticamente l'interfaccia utente. Attivando questo modalità si consiglia quindi di gestire anche l'evento *onSync (completed)* per aggiornare i dati a video.

❝ È ora possibile minimizzare le variazioni per ridurre la quantità dei dati scambiati tra client e server durante le sincronizzazioni differenziali. Per farlo è sufficiente chiamare il metodo *app.sync.DO.minimizeVariations.* Suggeriamo di chiamare il metodo in un batch notturno e comunque in assenza di altre sessioni utente o sessioni di sincronizzazione.

❝ È ora possibile verificare se i dati presenti sul database locale sono allineati a quelli del server. Per farlo è sufficiente chiamare il metodo *app.sync.DO.resyncAllClasses* con l'opzione *compareData*. Al termine dell'operazione tra le opzioni sarà restituita la collection *dataDiff* contenente tutte le differenze riscontrate:

- i documenti in stato deleted sono quelli presenti in locale ma non più presenti sul server;
- i documenti in stato updated sono quelli che avevano almeno un valore di proprietà diverso dal server;
- i documenti in stato inserted sono quelli che non erano presenti in locale ma erano presenti sul server.

WebAPI

❝ È ora possibile importare e consumare anche WebAPI in formato OData. Questo permette a due applicazioni Cloud di comunicare tra loro tramite WebAPI.

Ionic

❝ Gestione del tema scuro a livello globale: per attivare il tema scuro è possibile impostare la proprietà di tema *app.theme.darkMode* a *true* o *auto*.
È necessario inoltre rivedere le impostazioni di colore utilizzate dalla propria applicazione. Si consiglia di spostare tutte le impostazioni di colore dagli stili alle classi CSS in modo da poter utilizzare il selettore *.dark-mode* prima delle proprie classi. Tramite il medesimo selettore è possibile anche modificare le impostazioni predefinite.
Il test del tema scuro può essere effettuato anche con i nuovi controlli nell'IDE (nel popup di selezione Apple/Android) e nella preview (barra laterale destra).

Per maggiori informazioni sugli ulteriori parametri che regolano il tema scuro, puoi vedere la documentazione della proprietà *app.theme*.

❝ Gli elementi *IonCheckbox, IonRadio* e *IonToggle* hanno ora la proprietà "*label*" e "*itemSide*" per facilitarne l'utilizzo all'interno delle liste Ionic.

❝ L'elemento IonAutoComplete ha ora le proprietà "*allowNull*", che permette di svuotare il valore dell'elemento, "*comboClass*" per poter configurare la grafica della lista e "*dontClose*" per facilitare il debug. È inoltre stato specificato nella documentazione come utilizzare font icona di tipo diverso da Ionic per la grafica dell'applicazione.

❝ Ora è possibile aggiornare a runtime i colori del tema dell'applicazione, tramite il nuovo metodo *app.updateTheme*.

Bootstrap

❝ L'elemento *BSButton* ha ora le proprietà "*label*" e "*icon*" per facilitarne l'utilizzo. Non è più richiesto l'uso di un ulteriore elemento BSIcon per visualizzare un'icona.

Device

❝ Il plugin *app.device.preferences* ora espone nuovi metodi per controllare lo stato di ottimizzazione del consumo energetico su dispositivi Android di ultima generazione.

Miglioramenti

IDE

❝ Ora è possibile utilizzare il token "const" per definire una variabile.

APP

❝ Il metodo *app.open* ora permette di specificare anche l'opzione "*save*" per effettuare il download diretto di file memorizzati sul server.

❝ Il metodo *app.open* ora presenta nuove opzioni per permettere di scegliere l'applicazione con cui aprire il file su dispositivo mobile.

Documenti

❝ Se veniva chiamato il metodo *save* più volte sulla stessa istanza di documento l'operazione poteva generare errori. Ora la chiamata al metodo *save*, se è in corso una operazione di salvataggio, attende il completamento della prima operazione.

DataMap

❝ Se si passava *null* come secondo parametro del metodo *addFilter* di DataMap non veniva applicato il filtro "colonna is null".

❝ L'evento *datamap.onError* veniva notificato a volte con errore di tipo stringa, altre di tipo Error. Ora viene sempre notificato con errore di tipo Error.

Ionic

❝ Utilizzando il pulsante clear di una searchbar ora viene notificato solo l'evento *onClear* e non anche gli eventi *onBlur* e *onFocus* allo stesso tempo.

❝ È ora possibile utilizzare la proprietà "*tooltip*" per gestire i tooltip visuali degli elementi Ionic su device browser. Attenzione: l'attivazione di un tooltip su un elemento implica che esso acquisisce la proprietà CSS *overflow:visible*. Questo può causare problemi di visualizzazione che si possono risolvere solamente rimuovendo il tooltip.

❝ È stata aggiunta la proprietà *FilterText* all'oggetto *IonSelect*, che permette di personalizzare il testo mostrato nel placeholder del campo di ricerca della combo.

Cloud Connector

❝ Il Cloud Connector ora può essere utilizzato anche con Node 10.

Correzioni

IDE

❝ L'espressione *[var] is null* in una query produceva codice SQL invalido a run-time se la variabile utilizzata nell'espressione aveva valore NULL. A causa del malfunzionamento non veniva inserito uno spazio tra "null" (valore della variabile) e "is null" (espressione che seguiva la variabile nella query).

❝ Le liste valori non pubbliche non venivano definite a compile-time. Inoltre le proprietà di classe e applicazione, qualora fosse stato specificato un valore iniziale, non venivano inizializzate.

❝ Se si cambiava il tipo di una proprietà veniva eliminato l'evento *onChange* qualora presente. Inoltre se una proprietà contenente un evento *onChange* veniva cambiata in un metodo, l'evento *onChange*, qualora presente, non veniva eliminato ed era necessario rimuoverlo manualmente.

❝ Se si eliminava una foreign key il campo usato dalla relazione non veniva scollegato dal campo della tabella corrispondente. Per esempio: se nella TAB2 era presente una foreign key che collegava il campo FLD2 con il campo PK1 della TAB1 e si eliminava la relazione, il campo FLD2 rimaneva collegato al campo PK1 anche se la relazione era stata eliminata.

❝ Se si importava un componente si poteva ottenere l'errore "Non è permesso utilizzare YIELD se i metodi chiamati sono sconosciuti" in alcuni casi molto particolari. Il malfunzionamento si manifestava solo se:
- nel componente era presente una classe BASE contenente un metodo asincrono
- nel componente era presente una classe estesa EXT che estendeva la classe BASE
- nella classe estesa EXT era presente una chiamata al metodo della classe BASE
- il componente veniva importato in un progetto

In questo caso particolare veniva segnalato l'errore nella chiamata al metodo della classe BASE quando veniva avviata una qualunque applicazione contenuta nel progetto ove era stato importato il componente.
Malfunzionamento relativo alla richiesta di assistenza 000837-2019.

❝ Se si inserivano dei commenti nella select-list delle query della videata di esecuzione query dell'IDE, la tabella dei risultati conteneva colonne in più con intestazione "SELECT".

❝ Nei metodi non era possibile referenziare variabili locali all'interno delle espressioni di una sub-query.

❝ In un caso molto particolare una clausola where veniva tagliata. Il malfunzionamento si manifestava, per esempio, in questo caso:

```
...
where
 attivitaCliente.datafine between current_date - interval view.toDate and current_date
```

Dopo aver digitato il testo il Code Editor eliminava la parte finale "and current_date".

❝ La videata delle traduzioni mostrava anche le stringhe non traducibili dei componenti importati nel progetto. Ora queste stringhe vengono ignorate e non sono presenti nelle liste di stringhe da tradurre.

DataMap

❝ Se si impostava la collection di una datamap con documenti tutti invisibili non veniva notificato l'evento *onDataChange*.

❝ Poteva capitare di ottenere delle eccezioni di NullPointer se si accedeva ad elementi del template nell'evento *onRowComposition* dopo operazioni asincrone (operazioni precedute dal token *yield*).

❝ Se in una DataMap con query di caricamento si cambiava l'alias del campo auto-increment inserendo una nuova riga il valore del contatore non veniva riletto.

❝ Se all’interno dell'evento *onDataChange* venivano eseguite operazioni sul documento che causavano ulteriori modifiche, il sistema notificava nuovamente l'evento prima che terminasse l'esecuzione dell'evento precedente. Questo poteva causare sovrapposizioni di codice. Ora il sistema attende il completamento dell’evento *onDataChange* prima di notificarlo nuovamente.

Documenti

❝ Chiamando il metodo *restoreOriginal* su un documento con proprietà derivate, tali proprietà non venivano ricalcolate se le proprietà sorgenti erano modificate.

❝ Il servizio di lock ottimistico dei documenti non faceva fallire il salvataggio qualora non venisse aggiornato uno ed un solo record.

❝ Se si cambiava una proprietà *Blob* assegnando un *ArrayBuffer* il documento passava in stato modificato anche se il valore precedente aveva gli stessi byte.

SYNC

❝ In alcuni casi molto particolari il sistema inseriva il testo di errore "*WARN - Caught app exception: log.push is not a function - TypeError: log.push is not a function*" nei log del server. L'errore si poteva manifestare se era attivo il *performanceLog* (attivo per default) e se avveniva una disconnessione durante la sincronizzazione completa.
N.B.: l'errore non causava problemi alla procedura di sincronizzazione.

❝ Se nell'evento *onGetTopics* veniva generata un'eccezione, questa veniva gestita automaticamente dal sistema e non veniva segnalata nei file di log. Inoltre i topics rimanevano impostati al valore precedente all'eccezione. Ora le eccezioni generate all’interno dell’evento *onGetTopics* vengono correttamente segnalate al programmatore.

Ionic

❝ Il gesto "edgeSwipe", utilizzato solitamente per ritornare alla videata precedente al posto di premere il pulsante "back", ora chiama correttamente l'evento *onBackButton* invece che effettuare sempre la chiusura della videata.
Per la correzione del malfunzionamento è necessario aggiornare il pacchetto IonicV2 alla versione 19.5.

❝ L'altezza del componente *ionSelect* ora è corretta anche nel caso di *labelposition=stacked*.

❝ Su iOS 12.3 o successivi, effettuare uno swipe su un elemento *ionInput* che aveva il fuoco causava il riavvio dell'applicazione. Il workaround era impostare il parametro di tema: app.theme.dontHoldCaret = "true". Nella versione 19.5 questo errore è stato risolto.

❝ Su iOS le mappe di Google apparivano solo nella prima videata; nelle videate successive apparivano solo quando si interagiva con esse. Questo problema è stato risolto ritardando di 100ms l'impostazione del tipo di mappa rispetto all'apertura della videata.

❝ È stato corretto il funzionamento dei pulsanti +/- del tastierino numerico Ionic.

❝ L'elemento *IonAutoComplete* interpreta in maniera più estesa i nomi delle colonne dello schema della DataMap ad esso collegata. In particolare, sui nomi delle colonne convertiti in minuscolo vengono fatti i seguenti controlli:

1. Colonna del valore: ="id", ="code", ="value", ="v"
2. Colonne di decodifica: contiene "name", contiene "description" se non era già stata identificata, = "n"
3. Colonne di dettaglio: contiene "description" e non è stata usata per la decodifica, contiene "detail", ="d"
4. Colonna dell'icona: contiene "image", contiene "icon", ="src"
5. Colonna dello stile CSS o della classe CSS: ="style", ="class", ="s"
6. Se nessun altro controllo ha avuto successo, come valore viene considerata la prima colonna e come decodifica la seconda.

Bootstrap

❝ La proprietà cmdKey di pulsanti e input non aveva effetto all'interno di videate modali.

APP

❝ Le regole ACS non venivano applicate se l'applicazione usava i Proxy per aggiornare la UI.

❝ Ora gli elementi Input di tipo Radio comunicano al server il valore della proprietà *checked*. Durante l'evento *onChange* tutte le proprietà *checked* dei radio dello stesso gruppo sono già aggiornate.

❝ L'elemento Input di tipo "datetime" o "datetime-local" collegato ad un documento di una DataMap adesso permette di scrivere correttamente anche la parte dell’ora senza svuotarsi.

❝ Le applicazioni avviate dall'IDE non potevano modificare file o directory creati dalla console. Allo stesso modo file e directory creati da applicazioni avviate dall'IDE non potevano essere eliminati dalla console.
Malfunzionamento relativo alla richiesta di assistenza 000852-2019.

❝ Se il progetto conteneva risorse di tipo JS-SERVER si potevano ottenere errori a run-time se l'applicazione veniva avviata in modalità offline. Il malfunzionamento si manifestava sia se l'applicazione veniva avviata in modalità offline dall'IDE sia se veniva installata all'interno di un launcher su dispositivo.

❝ Nelle versioni precedenti se si scriveva

```
var s = (0.5).toLocaleString('it-IT')    ->          0.5
var d = new Date().toLocaleString('it-IT')       ->          05/14/2019 09:15:30 AM
```

si otteneva un risultato non corretto se il server (IDE o PROD) utilizzava Docker. Questo perché la versione di Node per Docker è “minimal” e non contiene i file per la localizzazione. Per ottenere questa correzione è necessario aggiornare il sistema operativo dei server IDE e di produzione, come indicato nell’introduzione di questo documento.

❝ Il plugin Facebook per browser è stato aggiornato ai nuovi livelli di API di Facebook e ora funziona correttamente. Il problema era presente solo nella versione browser e non tramite plugin nativo in-app.

❝ Il modulo peerjs non è compatibile con la nuova versione di socket.io utilizzata da Instant Developer Cloud 19.5. Pertanto WebRTC, utilizzato dall'IDE e dalle applicazioni installate, potrebbe non funzionare correttamente.

WebAPI

❝ Le chiamate WebAPI a classi definite direttamente nell'applicazione generavano eccezione: "*Cannot read property 'webApiService of undefined*".

❝ Se si attivavano le WebAPI su una classe con una proprietà collegata ad una lista valori definita in una tabella di database, si otteneva errore durante la pubblicazione. Per aggirare il problema è sufficiente spostare la lista valori fuori dalla tabella.

Cloud Connector

❝ Se avveniva una disconnessione durante un'operazione di *put*, i file lato Cloud Connector rimanevano aperti.

MyCloud

❝ In alcuni casi Node.js si riavviava ogni 15 minuti se si installavano una o più applicazioni su un server MyCloud.