Guida ai componenti

Pro Gamma Instant Developer

Il sistema più semplice e veloce per sviluppare
Rich Internet Application di classe Enterprise

Instant Developer: guida ai componenti

Sommario

Capitolo 1

Component Gallery

1.1 Introduzione alla gallery

Le funzionalità descritte nella guida all'uso di Instant Developer possono essere applicate allo sviluppo di componenti specifici per le proprie applicazioni, ma anche alla realizzazione di componenti di uso generale riutilizzabili in tanti contesti diversi.

La component gallery è un raccoglitore di componenti ad uso generico, ora realizzati da Pro Gamma, ma aperta anche ad altri produttori. Alcuni componenti sono gratuiti e, oltre a fornire utili funzioni, servono anche come esempio di utilizzo. Quelli più complessi richiedono il pagamento di un prezzo perché è stato impiegato un notevole tempo di sviluppo o perché hanno richiesto particolari conoscenze tecnologiche non facilmente reperibili sul mercato.

Tutti i componenti hanno una versione dimostrativa senza scadenza temporale, che permette di integrarli nelle proprie applicazioni, controllarne il corretto funzionamento e proporli agli utenti finali per verificarne il gradimento; in questo modo si possono acquistare con la sicurezza di poterne ricavare un valore aggiunto. L'unica limitazione della versione dimostrativa consiste nel fatto che anche l'applicazione in cui vengono usati verrà compilata in modalità demo.

I componenti in versione non dimostrativa contengono anche i sorgenti, cioè il progetto Instant Developer completo. In questo modo è possibile vedere come particolari funzioni sono state realizzate, correggere eventuali malfunzionamenti o, infine, adattarli a specifiche esigenze.

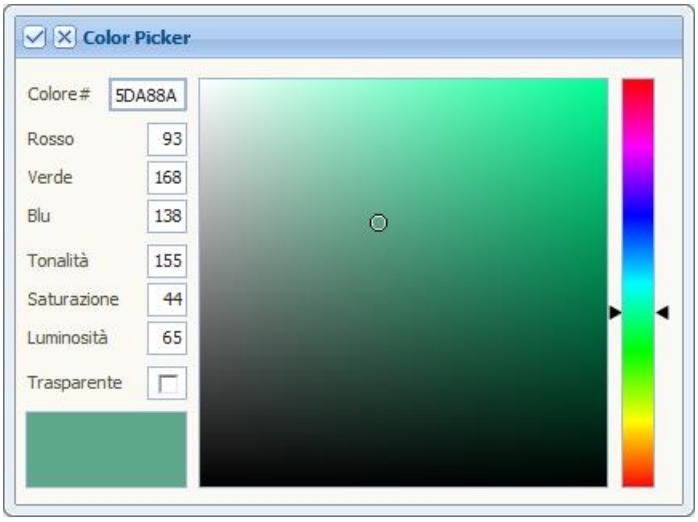
Capitolo 2

Color Picker

2.1 Il componente Color Picker

Il componente Color Picker mette a disposizione una videata da aprire in popup da utilizzare per la scelta di un colore da parte dell'utente. Per includerlo nelle proprie applicazioni è sufficiente scaricare dalla component gallery il file *ColorPicker.idz*, importarlo nel progetto e poi seguire le istruzioni descritte nei paragrafi successivi.

Vediamo adesso come si presenta la videata di scelta del colore.



Videata Color Picker

L'utente può scegliere un colore in quattro modi:

1) cliccando sulla barra laterale per scegliere la tonalità e poi sulla paletta di colori per scegliere saturazione e luminosità.
2) digitando il valore esadecimale del colore nel campo *Colore#*;
3) digitando il valore delle componenti rosso, verde e blu;
4) digitando il valore della tonalità, saturazione e luminosità.

Appena l'utente modifica uno dei parametri l'interfaccia si aggiorna di conseguenza. Per confermare la scelta del colore, occorre premere il pulsante nella barra del titolo della videata.

2.2 Apertura del Color Picker

Per collegare l'apertura della videata ad un campo di pannello o box di report è sufficiente collegare ad essi una procedura come quella mostrata nell'immagine seguente. Il codice è molto semplice: dopo aver aperto la videata in popup, è possibile chiamare la procedura *SetColor* passando come parametro il numero intero corrispondente al colore desiderato. Si tenga presente che il colore trasparente equivale a -1.

```
public void ColorPickerApp.ApriColorPicker()
{
    ColorPicker.show(Popup)
    ColorPicker.SetColor(RGBColor(93, 168, 138, ...))
}
```

Esempio di codice per aprire il Color Picker e preimpostarne il colore

Per comporre un colore da codice si possono utilizzare le funzioni RGBColor e HSBColor. Se non viene impostato il colore iniziale, verrà mostrato il colore bianco.

2.3. Lettura del colore scelto dall'utente

Quando l'utente conferma la scelta del colore premendo il tasto nella barra del titolo, il codice del componente invia un messaggio alla videata da cui è stato aperto per comunicare la scelta dell'utente.

Nell'evento OnSendMessage della videata da cui è stato aperto il Color Picker, si può recuperare il colore scelto dall'utente che viene passato come numero intero nel parametro *Par1*.

```
event Videata.OnSendMessage(
   string Message // Indica il nome del messaggio
   IDForm Sender  // Identifica la form che ha inviato il messaggio
   IDDocument Document // Optional document sent by the sender
   string Par1    // Opzionale. Contiene una espressione qualsiasi (il cui ...
   string Par2    // Opzionale. Contiene una espressione qualsiasi (il cui ...
   string Par3    // Opzionale. Contiene una espressione qualsiasi (il cui ...
   string Par4    // Opzionale. Contiene una espressione qualsiasi (il cui ...
)
{
  // Se il messaggio proviene dal ColorPicker
  if (ColorPicker.isMyInstance(Sender))
  {
    if (Message == "SetColor")
    {
      // Imposto lo sfondo di un campo con il colore scelto dall'utente
      Pannello.Campo.backgroundColor = toInteger(Par1)
    }
  }
}
```

Recupero del colore scelto dall'utente

Nell'esempio il colore scelto viene utilizzato per cambiare il colore di sfondo di un campo del pannello. Analogamente è possibile modificare i colori dello sfondo e del testo di box, span e campi di pannello impostando le proprietà BackgroundColor e TextColor.

La nuova libreria IDImage, oltre ai metodi per la lettura, scrittura e modifica delle immagini, contiene funzioni per disegnare linee, rettangoli, ellissi e testo. Il Color Picker può essere usato per chiedere all'utente con quale colore vuole disegnare, come è esemplificato nell'esempio Immagini dell'application gallery.

Capitolo 3

Visual Query Builder

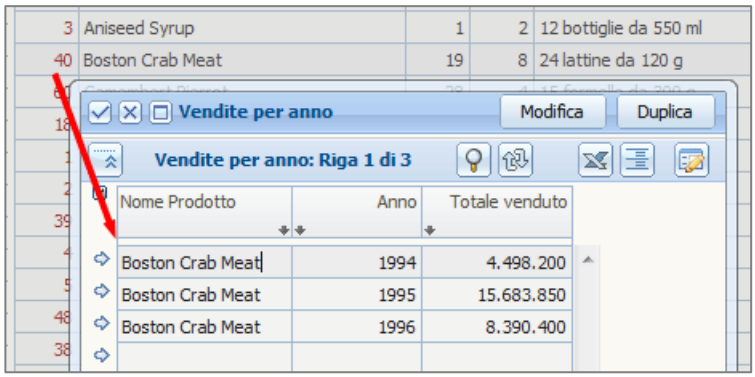
3.1 Visual Query Builder (VQB)

Normalmente un'applicazione software permette di gestire alcune tipologie di dati all'interno di un processo operativo adeguato. Quando questo processo non è elementare e quando il numero degli utilizzatori aumenta, non è facile prevedere tutte le possibili modalità di utilizzo e di visualizzazione dei dati, che spesso variano da utilizzatore a utilizzatore.

Per risolvere questo problema sarebbe sicuramente utile poter definire direttamente a runtime delle nuove videate di interrogazione e di selezione dati, senza bisogno di modificare il codice dell'applicazione. L'ideale sarebbe se questa operazione potesse essere compiuta direttamente dall'utente finale in completa autonomia.

Il componente Visual Query Builder, di seguito abbreviato con VQB, nasce per rispondere a questa esigenza. Le caratteristiche principali sono le seguenti:

1) Permette di creare direttamente a runtime videate contenenti un pannello di selezione sui dati dei database a cui l'applicazione si può collegare.
2) Le query di selezione vengono composte in modo quasi totalmente visuale. Anche un utente non particolarmente esperto può riuscire ad estrarre i dati di interesse.
3) Le videate sono graficamente configurabili. E' possibile modificare i colori del testo, dello sfondo, la larghezza e l'ordine dei campi, il tipo di carattere e così via.
4) La definizione delle videate viene salvata in formato XML, quindi non richiede un ulteriore database. E' tuttavia possibile configurare il componente per salvare la definizione anche nelle tabelle.
5) Le videate definite possono essere collegate ad altre, in modo da estrarre dati contestualizzati, oppure per recuperare dati da riportare nei pannelli sottostanti.
6) I dati presenti nelle videate possono essere raggruppati, totalizzati ed esportati.
7) La definizione delle videate può essere condivisa fra più utenti in base al loro ruolo.
8) Esiste un sistema di profilazione che permette di configurare i dati che ciascun utente può utilizzare all'interno delle videate personalizzate.

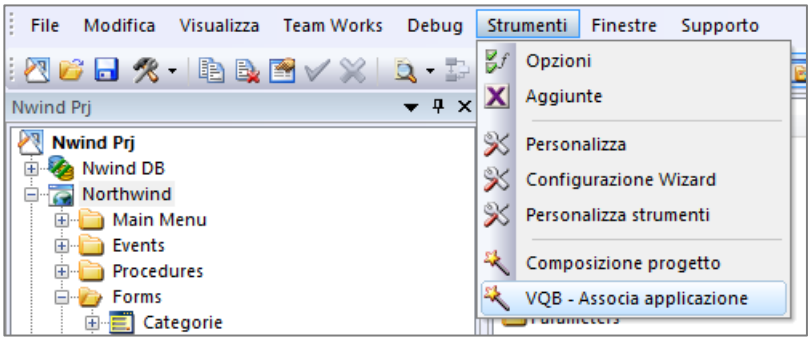


Esempio di videata personalizzata che mostra le vendite di un prodotto

3.2 Integrazione con il progetto

Vediamo ora i passi necessari da compiere per integrare VQB all'interno di un proprio progetto. L'operazione è molto semplice e non richiede più di un'ora di lavoro.

Il primo passaggio consiste nel creare un file di descrizione della struttura del database, in modo che VQB possa mostrarla all'utente quando definisce i dati da selezionare nelle videate. Questa operazione viene svolta dall'interno dell'IDE di Instant Developer tramite il comando *VQB – Associa applicazione* presenti nel menù *Strumenti*.



Questo comando apre una videata in cui vengono elencate le applicazioni del progetto. Premendo il bottone *Associa* sulla destra delle stesse, verrà creato un file XML che descriverà i database presenti nel progetto. Il file verrà poi salvato nella cartella *custom* dell'applicazione che sarà creata se non ancora esistente. In questo modo diventerà parte dell'applicazione e sarà presente anche nel server web in modo da poter essere caricato a runtime dal componente VQB.

Il file XML descrive le tabelle e le viste presenti in tutti i database del progetto. Se si desidera escluderne qualcuna, è possibile creare ed attivare un'area tematica prima di premere *Associa*. Premendo più volte il pulsante, il file verrà aggiornato.

A questo punto occorre importare il componente nel progetto. La versione dimostrativa può essere scaricata direttamente dalla Component Gallery del sito di Instant Developer. Si consiglia di salvare il file del componente (*VisualQueryBuilder.idz*) nella stessa directory in cui è presente il progetto; in questo modo quando si usa il comando *Importa Componente* nel menù contestuale del progetto, VQB apparirà subito nell'elenco di quelli disponibili.

Il terzo passaggio consiste nell'indicare al componente il ruolo dell'utente che ha attivato la sessione di lavoro. Vedremo successivamente che, grazie alla definizione dei ruoli, si può limitare la visibilità delle tabelle e dei campi e definire filtri specifici per le query. L'assegnazione del ruolo normalmente avviene nell'evento OnLogin, ma può essere effettuata anche in altri punti del codice; di solito avviene tramite i seguenti passaggi:

1) Si imposta la proprietà *Ruolo* di *VisualQueryBuilder* ad un valore stringa che verrà usato all'interno della configurazione del componente.
2) Se si desidera abilitare tutte le funzioni, è possibile impostare a *true* la proprietà *Amministratore*.
3) Infine è possibile impostare parametri globali da utilizzare nelle query inserendoli nella tabella in memoria relativa.

Nell'immagine seguente vengono esemplificati i passaggi descritti.

```
event Northwind.OnLogin(
  inout string Username   // E' una stringa co
  inout string Password   // E' una stringa co
  inout boolean DataValid // Se impostato a Tr
)
{
  if (Username == "ADMIN")
    VisualQueryBuilder.Amministratore = true
  else
  {
    VisualQueryBuilder.Ruolo = "ruolo1"
    //
    // Inserisco alcuni parametri globali
    // per il componente VQB
    insert values into ParametriGlobali
      set Nome = "IDUtente"
      set Valore = Username
    //
    insert values into ParametriGlobali
      set Nome = "IDAzienda"
      set Valore = 1
  }
}
```

Attribuzione del ruolo utente e passaggio di parametri

L'amministratore ha il compito di configurare i ruoli degli altri utenti ed ha pieni poteri sulle videate personalizzate già create. Un utente non amministratore, invece, può creare nuove videate basate solo sulle tabelle e campi visibili in funzione dal proprio ruolo, e può utilizzare solo le proprie videate o le videate pubbliche create da altri.

L'amministratore può utilizzare i record inseriti nella tabella *ParametriGlobali* per impostare criteri di filtro fissi su alcuni campi. Ad esempio, se si desidera che un utente possa vedere solo i propri ordini e non quelli degli altri, l'amministratore può fissare un criterio di filtro sul campo *Impiegato* della tabella *Ordini*, richiedendo che sia uguale al parametro *IDUtente* del componente.

L'ultimo passaggio consiste nell'inserire i comandi per accedere ad alcune funzioni del componente e per permettere agli utenti di creare nuove videate o utilizzare quelle esistenti. Le videate che si possono inserire nel menù principale sono le seguenti:

1) *ConfigurazioneRuoli:* utilizzata solo dall'amministratore per configurare i ruoli e gli accessi ai dati.
2) *ElencoVideatePersonalizzate:* contiene un elenco di videate che l'utente può utilizzare e che non necessitano di essere attivate a partire da un'altra videata presente nell'applicazione.

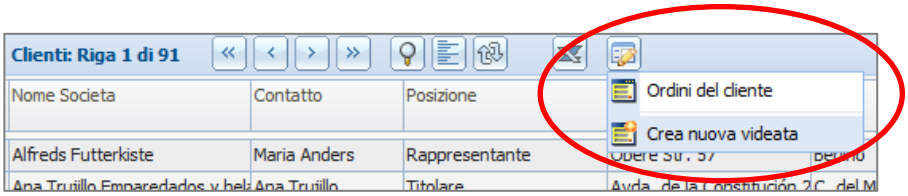
Per fare in modo che l'utente possa creare nuove videate a partire da altre già aperte, è possibile aggiungere alla toolbar di tutti i pannelli dell'applicazione un pulsante perso-

nalizzato. Esso viene gestito rendendo globale l'evento OnCommand dei pannelli, come mostrato nel codice sottostante, che può essere usato così com'è nei propri progetti.

```
event Northwind.Initialize()
{
  Northwind.initCustomCommand(Custom1, "Apri videate collegate", "customform"
    , "Mostra l'elenco delle videate personalizzate", 0)
}
event Northwind.GlobalPanelCommand(
  IDPanel Panel          // Oggetto che notifica questo evento
  int Command            // E' un numero intero che rappresenta il comando ...
  inout boolean Cancel   // Può essere impostato a True per cancellare il c...
  boolean UserOperation // E' un valore booleano che vale True se l'evento...
)
{
  switch (Command)
  {
    case Custom1:
      VisualQueryBuilder.FormsManager.OpenFormsMenuPopup(Panel.parentForm()
        , Panel.getRD3ID(ToolbarButton, Custom1), Modal)
    break
  }
}
```

Sopra: aggiunge il pulsante; sotto: apre menù contestuale.

Il risultato è mostrato nell'immagine sottostante:



Ora è possibile compilare l'applicazione. Dopo che l'amministratore avrà configurato i ruoli, gli utenti potranno realizzare le videate di loro interesse senza richiedere modifiche al codice dell'applicazione.

3.3 Oggetti principali di VQB

Prima di entrare nel dettaglio delle funzionalità di VQB, vediamo quali oggetti vengono messi a disposizione del programmatore e quali sono i principali metodi supportati:

DBManager	Classe addetta al caricamento della struttura dei database e delle funzioni di libreria, usata durante l'editing delle videate.

GetDatabases	Metodo che restituisce l'elenco dei database consultabili dall'utente, già filtrati in base al ruolo. La prima volta esegue anche il caricamento della struttura dal file *VQB_Databases.xml* generato dal sistema di associazione descritto nei paragrafi precedenti.
GetFunctions	Metodo che restituisce l'elenco delle funzioni utilizzabili dall'utente durante l'editing delle query. La prima volta esegue il caricamento dal file *VQB_Functions.xml* messo a disposizione dal componente stesso.

RolesManager	Classe addetta al caricamento dei ruoli utente.
GetRoles	Metodo che restituisce l'elenco dei ruoli definiti dall'amministratore; la prima volta che viene chiamato essi vengono anche caricati. Solitamente i ruoli vengono memorizzati nel file *VQB_Roles.xml* generato dal metodo *SaveRoles*
SaveRoles	Metodo per salvare la definizione dei ruoli nel file *VQB_Roles.xml*

FormsManager	Classe addetta alla gestione delle videate personalizzate.
GetForms	Metodo che restituisce l'elenco delle videate che l'utente può utilizzare già filtrate in base al suo. La definizione di ogni videata è contenuta in un file di nome *VQB_Form-_{GUID}.xml* contenuta in una sotto directory dell'applicazione.
GetLinkedForms	Metodo che restituisce le videate personalizzate collegate alla form passata come parametro.
OpenFormMenuPopup	Metodo che apre il menu popup che contiene i comandi per l'utilizzo delle videate personalizzate a partire dalla form passata come parametro.
LoadForm	Metodo per caricare una videata.
OpenForm	Metodo per aprire, creare, modificare, duplicare una videata.
DeleteForm	Metodo per eliminare una videata.
SaveForm	Metodo per salvare una videata.
CloneForm	Metodo per clonare una videata.

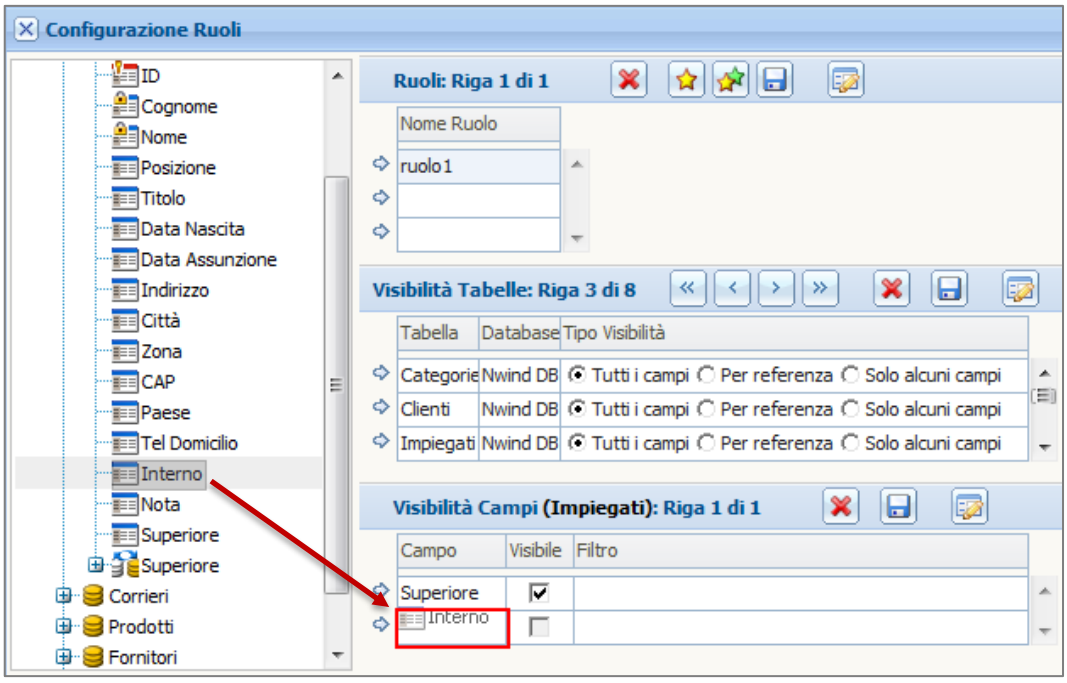
ConfigurazioneRuoli	Videata per la configurazione dei ruoli
ElencoVideatePersonalizzate	Videata per la gestione delle videate personalizzate che non devono essere aperte a partire da una specifica videata.
MyForm	Prototipo di form che viene mostrata quando l'utente apre una videata personalizzata.
ConfiguratoreVideata	Permette la configurazione dei dati visibili in una videate personalizzata.

3.4 Configurazione dei ruoli

Il componente VQB permette di definire un insieme di ruoli utente allo scopo di limitare la visibilità delle tabelle e dei campi su cui costruire le videate personalizzate. Nell'immagine sottostante viene mostrata un'anteprima della videata *ConfigurazioneRuoli* tramite la quale l'amministratore potrà effettuare questa operazione.

Sulla sinistra è presente una visualizzazione ad albero che contiene l'elenco delle tabelle disponibili, mentre a destra ci sono i pannelli che contengono i dati dei ruoli.

Il primo pannello in alto, *Ruoli*, contiene semplicemente i ruoli definiti. La proprietà *Ruolo* vista in precedenza va impostata ad uno dei valori presenti in questo pannello.



Anteprima della videata per la configurazione dei ruoli

Selezionando un ruolo nel primo pannello, nell'elenco *Visibilità Tabelle* appariranno le tabelle che esso può consultare. Per aggiungere una tabella è possibile tirarla con il mouse dall'albero nel pannello; se essa contiene delle relazioni, le tabelle correlate verranno automaticamente aggiunte per referenza.

Il campo *Tipo Visibilità* permette di specificare quali campi possono essere letti. I possono valori sono i seguenti:

1) *Tutti i campi:* l'utente potrà vedere tutti i campi della tabella.
2) *Per referenza:* l'utente potrà vedere solo i campi chiave e quelli descrittivi. Viene usato per poter sapere quale record viene referenziato da un'altra tabella in relazione con essa.
3) *Solo alcuni campi:* L'utente potrà vedere solo i campi elencati nel pannello *Visibilità Campi* più in basso.

Per aggiungere un campo a quest'ultimo elenco è possibile tirarlo con il mouse dall'albero nel pannello. E' anche possibile definire un'espressione di filtro per il campo, che verrà automaticamente aggiunta a tutte le query che lo coinvolgono.

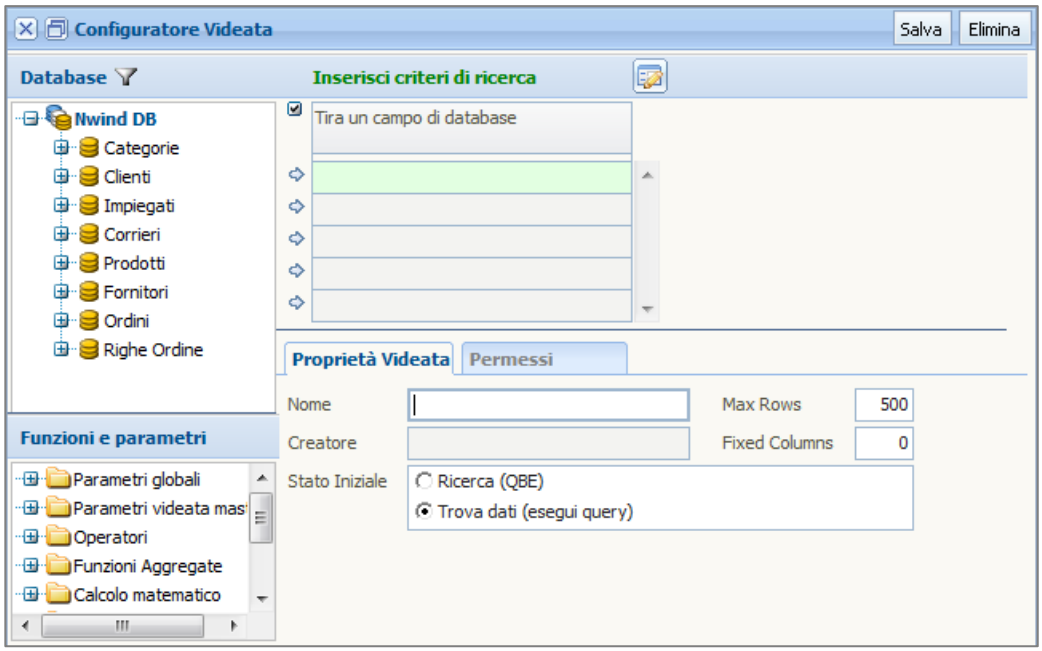
L'espressione del filtro va scritta omettendo il nome del campo: se, ad esempio, si vuole filtrare il campo *IDAzienda* selezionando solo il valore 1 si deve scrivere "=1". E' possibile anche utilizzare i parametri globali mettendo il nome del parametro fra parentesi graffe, come ad esempio "={IDAzienda}".

3.5 Creazione di una videata personalizzata

Per definire una nuova videata occorre aprire il *Configuratore Videata*, attraverso i modi previsti nei progetto. Se si integra il componente come descritto nel paragrafo precedente, questo può avvenire con le seguenti operazioni:

1) Aprendo la videata *ElencoVideatePersonalizzate* in cui è presente il pulsante *Crea nuova videata*; in tal caso la nuova videata definita potrà essere aperta in ogni momento e non solo a partire da una specifica videata.
2) Tramite il menu popup aperto dal pulsante personalizzato nella toolbar di un pannello; in questo caso la nuova videata sarà collegata a quella di partenza, potendone referenziare i campi ed impostarne i valori.

Nell'immagine seguente viene mostrato il configuratore di videate:

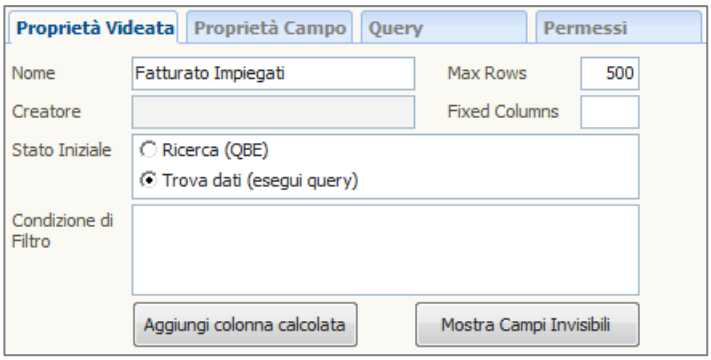


Definizione delle proprietà della videata

Le proprietà che si possono definire nella scheda relativa sono le seguenti:

1) *Nome*: è il nome della videata usato nei menù e nell'intestazione della stessa.
2) *Stato iniziale*: permette di scegliere se la query deve essere eseguita subito all'apertura della videata oppure si devono prima inserire i criteri di ricerca.
3) *Max Rows*: numero massimo di righe che verranno recuperate dal database e mostrate nel pannello. Se viene impostato a zero non ci sarà alcun limite, ma non è consigliabile farlo perché una query con un risultato troppo grande può richiedere molta memoria nel server.
4) *Fixed Columns*: numero di colonne che devono rimanere visibili quando la videata ha la scrollbar orizzontale; se impostato a zero non ci saranno colonne bloccate.
5) *Condizione di filtro*: condizione aggiuntiva di filtro per i dati della videata.

Solo l'amministratore può accedere alla pagina *Query*, in cui si può vedere e modificare il testo SQL. Normalmente non è necessario farlo perché sono sufficiente le operazioni visuali che verranno descritte nei paragrafi seguenti.



Anteprima del pannello Proprietà Videata

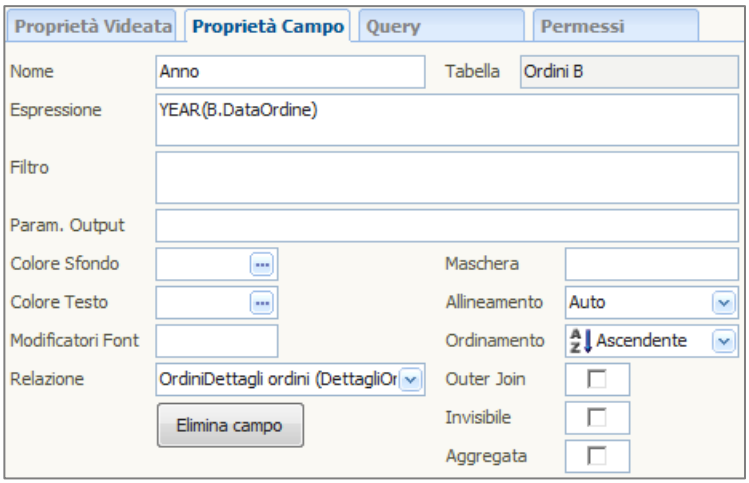
Selezione dei campi da visualizzare

Per aggiungere un campo alla videata è sufficiente tirarlo dall'albero nel pannello in alto. Se si desidera aggiungere un'espressione calcolata senza partire da un campo è possibile premere il pulsante *Aggiungi Espressione*.

Se si desidera avere un'anteprima dei dati contenuti in una tabella, è possibile tirarla con il mouse sulle linguette, oppure utilizzare il comando di menù contestuale *Mostra dati*. Verrà aperta una piccola griglia di dati che ne mostrerà il contenuto e sarà possibile tirarne una colonna nel pannello in alto per aggiungerla alla selezione.

Quando si aggiunge al pannello il primo campo, di fatto si sceglie anche la tabella che lo contiene. A questo punto nell'elenco delle tabelle verranno mostrato solo quelle correlate con la prima, in modo da essere più guidati nella scelta dei campi ulteriori. Se si desidera vedere tutte le tabelle è possibile cliccare il pulsante nella barra del titolo della visualizzazione ad albero.

Cliccando in un campo selezionato del pannello, è possibile modificarne le seguenti proprietà:



Anteprima del pannello in cui si specificano le proprietà di un campo

1) *Nome*: intestazione del campo nel pannello.
2) *Espressione*: è il testo dell'espressione, espresso in codice SQL. Quando il campo è stato aggiunto tirando con il mouse gli oggetti dall'albero, il valore dell'espressione viene preimpostato al nome fisico del campo nel database.
3) *Filtro*: condizione di filtro sul campo. Non si deve specificare il nome del campo, ma solo la condizione. Ad esempio, se si vuole filtrare il campo *IDCategoria* con il valore 1, occorre scrivere solo "=1".
4) *Parametro output*: questa proprietà deve essere impostata solo se si desidera che il valore venga riportato nel pannello sottostante. In questo caso è possibile tirare uno dei parametri della videata master all'interno di questa proprietà.
5) *Colore di sfondo*: colore dello sfondo del campo. Va espresso in formato esadecimale, ad esempio *FF02B2*. Cliccando sul pulsante sulla destra del campo è possibile selezionare il colore tramite una videata dedicata (ColorPicker).
6) *Colore del testo*: colore del testo del campo.
7) *Maschera*: indica come deve essere mostrato il valore. Il formato di questa proprietà corrisponde a quello della proprietà Mask dei campi di pannello.
8) *Allineamento*: indica dove deve essere mostrato il valore: a sinistra, centrato, a destra o automatico.
9) *Modificatori font*: indica il tipo di carattere da utilizzare. Possono essere inseriti i seguenti valori, anche più di uno: B=grassetto, I=italico, U=sottolineato, S=barrato.
10) *Ordinamento*: tipo di ordinamento dei dati, ascendente o discendente.
11) *Relazione* condizione di join tra la tabella a cui appartiene il campo e un'altra tabella coinvolta nella query. Può essere cambiata se ci sono più relazioni che coinvolgono il campo fra le tabelle selezionate nella query.
12) *Outer join*: se attivato, il legame tra le due tabelle sarà di tipo opzionale.

13) *Invisibile*: se viene deselezionato, il campo non verrà mostrato quando la videata sarà utilizzata normalmente. In questo caso i dati presenti nel panello possono cambiare se esso contiene espressioni aggregate in quanto un campo non visibile non influisce sui raggruppamenti. Per vedere il risultato finale è possibile premere il pulsante *Mostra Campi Invisibili* o *Nascondi Campi Invisibili* nella scheda delle proprietà della videata.
14) *Aggregata*: indica che l'espressione utilizza una funzione aggregata. Normalmente viene attivato automaticamente quanto si tira dall'albero una funzione aggregata.
15) *Raggruppamento*: seleziona il tipo di totalizzazione quando i dati del pannello vengono mostrati in forma raggruppata, ad esempio *Somma* o *Conteggio*.

L'espressione del campo può essere scritta direttamente nella scheda delle proprietà, oppure si possono tirare con il mouse oggetti dall'albero, sia sul testo dell'espressione che sulla colonna corrispondente nel pannello di anteprima dei dati. Infine, le seguenti operazioni possono essere effettuate direttamente sul pannello di anteprima:

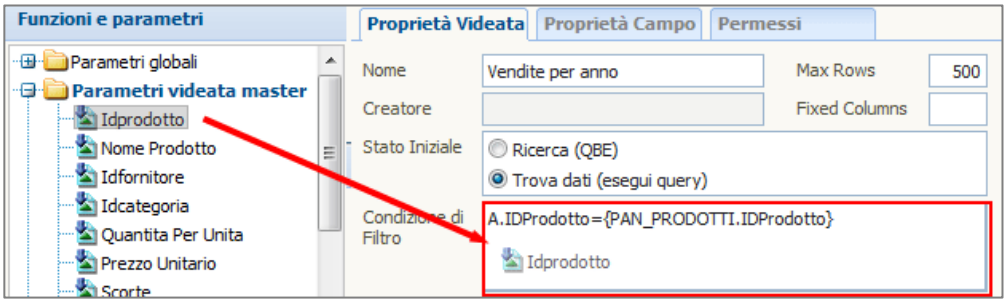
1) Cliccando sulle intestazioni delle colonne si modifica l'ordinamento del campo corrispondente.
2) Allargando l'intestazione con il mouse si ridimensiona la colonna.
3) Trascinando una cella su un'altra si modifica l'ordine delle colonne.

Collegamento con la videata master

Quando la nuova videata viene creata a partire da una esistente, è possibile riferire i valori contenuti in quella di partenza, oppure riportare dati in essa.

La prima operazione si ottiene creando filtri sui campi o sull'intero pannello usando gli oggetti dell'albero contenuti nella sezione *Parametri videata master*. In essa vengono elencati i campi dei pannelli della videata di partenza.

Se, invece, si vuole utilizzare la videata personalizzata come finestra di ricerca, occorre tirare dall'albero il campo che si vuole valorizzare nella proprietà *Parametro Output* della scheda delle proprietà del campo.



Aggiungere un filtro in base al prodotto scelto nella videata di partenza

Definizione dei permessi di utilizzo

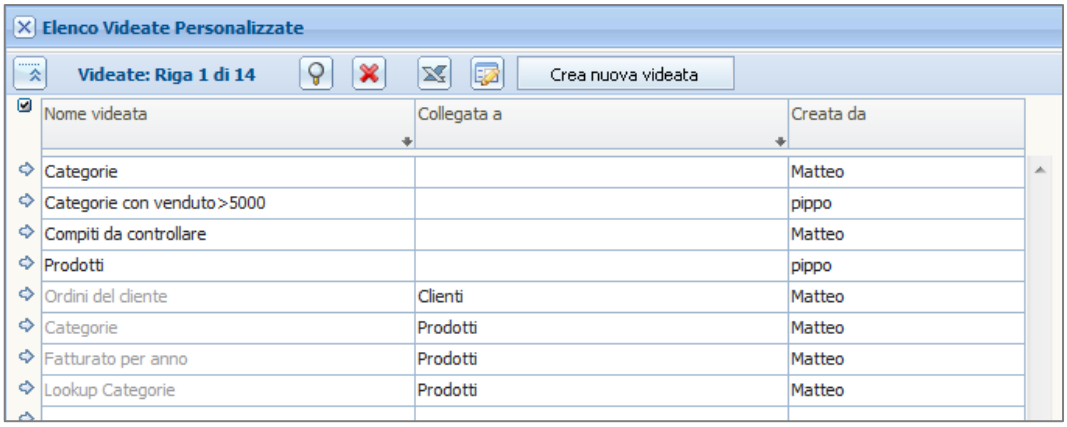
Se non viene indicato nessun permesso, solo chi ha creato la videata può utilizzarla. Nella scheda *Permessi* è allora possibile definire quali ruoli possono utilizzarla. Per abilitare un ruolo è sufficiente spuntare il check-box sulla sinistra. Si ricorda che l'amministratore ha sempre accesso a tutte le videate create dagli utenti.

Anteprima del pannello Permessi

3.6 Uso delle videate personalizzate

Una volta create le proprie videate l'utente potrà riaprirle a seconda alle modalità che sono state previste in fase di sviluppo.

Si consiglia di rendere disponibile la videata *Elenco Videate Personalizzate* in cui vengono elencate tutte le videate disponibili. Quelle che non sono legate ad altre videate potranno essere aperte dalla lista. E' disponibile anche un pulsante per creare una nuova videata non collegata ad altre.



Anteprima dell'elenco Videate Personalizzate

Una volta aperta una videata personalizzata, chi l'ha creata o l'amministratore può modificarla. In questo caso, infatti, appariranno i pulsanti relativi nella barra del titolo.

Anteprima dei comandi per modificare la videata personalizzata

Come in qualunque altro pannello, nelle videate personalizzate l'utente può:

1) Utilizzare le funzioni query-by-example per inserire criteri di ricerca ulteriori.
2) Ordinare o raggruppare i dati cliccando sull'intestazione dei campi.
3) Esportare i dati in formato excel.

3.7 Personalizzazione del componente

La versione acquistabile del componente VQB contiene anche i sorgenti per consentire di modificare, adattare o estendere il componente alle proprie esigenze se lo si ritiene necessario.

Per cambiare i comportamenti del componente è però possibile agire per estensione invece che modificando il codice. Ad esempio, se si preferisse salvare le videate nel database invece che nei file xml, si potrebbe creare una classe all'interno della propria applicazione che estende *FormsManager* del componente. Questa classe potrebbe ad esempio essere chiamata *MyFormsManager*.

A questo punto si dovrebbero implementare i metodi *GetForms*, *LoadForm*, *DeleteForm*, *SaveForm* e *CloneForm*, secondo le proprie specifiche. In questa fase può essere utile avere il codice sorgente dei metodi originari per vedere nel dettaglio quali operazioni vengono eseguite.

Infine, nell'evento Initialize dell'applicazione si dovrebbe impostare la proprietà *FormsManager* del componente a quella della propria classe estesa, come mostrato nell'immagine seguente.

```
event Northwind.Initialize()
{
  MyFormsManager mfm = new()
  VisualQueryBuilder.FormsManager = mfm
}
```

Esempio di personalizzazione della classe FormsManager

Capitolo 4

Mappe di Google (GMAP)

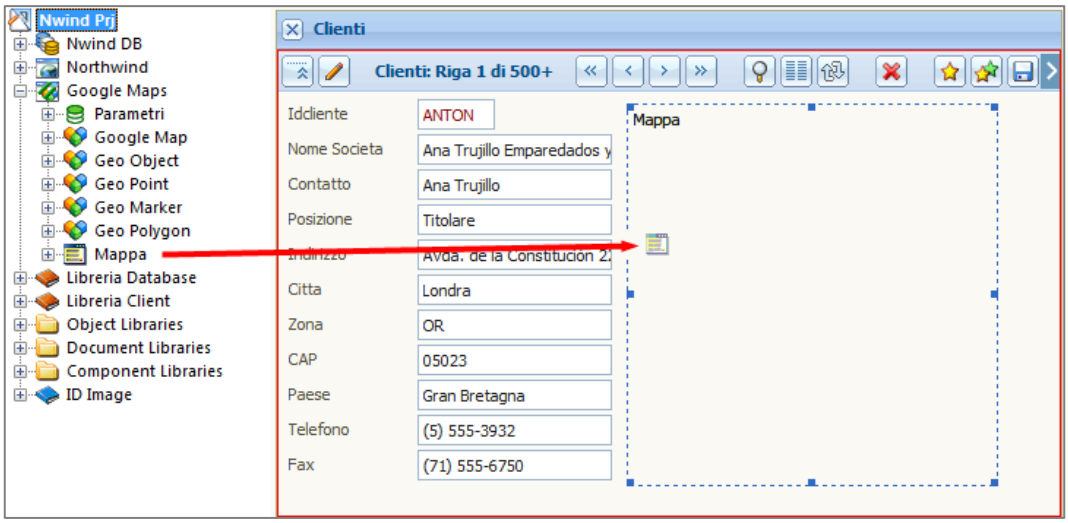
4.1 Il Componente GMAP

Il componente GMAP permette di utilizzare le funzionalità delle mappe di Google all'interno delle applicazioni realizzate con In.de. I servizi forniti da Google sono gratuiti, ma con alcune limitazioni; prima di utilizzare questo componente occorre aver letto ed accettato i termini di utilizzo. Le funzionalità del componente sono le seguenti:

1) Visualizzazione di mappe altamente configurabili all'interno delle proprie videate.
2) Posizionamento di marker e poligoni sulle mappe.
3) Notifica di eventi all'applicazione causati dall'interazione dell'utente.
4) Geo-decodifica di indirizzi in coordinate geografiche e viceversa.
5) Calcolo di percorsi (dalla versione 10.1)
6) Gestione della licenza Premier.

4.2 Integrazione del componente nel progetto

L'integrazione è immediata: dopo aver importato il componente nel progetto utilizzando il file *GoogleMaps.idz*, per mostrare una mappa in una videata basta tirare la videata Mappa, presente nel componente, su un campo statico, in una tabbed view o su un frame vuoto.



Un drag & drop per aggiungere la posizione del cliente nella mappa

4.3 Configurazione della una mappa

Dopo aver creato la sotto-videata della mappa occorre impostarne le proprietà. Quelle disponibili solo le seguenti:

ZoomLevel	Livello di zoom: da 0, tutto il mondo, a 21 in cui si vedono le singole costruzioni.
CenterLatitude	Latitudine su cui è centrata la mappa, in gradi decimali.
CenterLongitude	Longitudine su cui è centrata la mappa, in gradi decimali.
MapType	Tipo di mappa mostrata. Il valore predefinito è *Roadmap* ed i possibili valori sono elencati nella lista valori *MapTypes*.
DoubleClickZoom	Indica se il doppio click sulla mappa permette di cambiare lo zoom e le coordinate del centro. Il valore predefinito è *true*.
ScrollWheel	Indica se l'utilizzo della rotella del mouse permette di cambiare lo zoom. Il valore predefinito è *true*.
Draggable	Indica se è ammesso il trascinamento della mappa per cambiare le coordinate del centro. Il valore predefinito è *true*.
KeyboardShortcuts	Indica se è ammesso usare i tasti per interagire con la mappa. Il valore predefinito è *true*.
MapTypeControl	Indica se è visibile il controllo per cambiare il tipo di mappa. Il

	valore predefinito è *true*.
ScaleControl	Indica se è visibile il controllo con la scala di misura. Il valore predefinito è *true*.
StreetViewControl	Indica se è visibile il controllo per passare alla visualizzazione StreetView. Il valore predefinito è *true*.
AutoZoom	Dalla versione 10.1, indica che la mappa si adatta automaticamente ai marker che contiene. Il valore di default è *true*.

Se, ad esempio, si vuole visualizzare una cartina dell'Italia occorre scrivere il seguente codice:

```
event Videata.Load()
{
  Mappa mapSF = Mappa.IDForm()
  GoogleMap map = mapSF.GetMapOptions()
  //
  // Visualizzo l'Italia
  map.ZoomLevel = 6
  map.CenterLatitude = 42,75
  map.CenterLongitude = 12,49
  //
  // Applico le modifiche
  map.UpdateMap()
}
```

4.4 Posizionamento di oggetti sulla mappa

Nella versione attuale del componente è possibile visualizzare sulla mappa dei marker e dei poligoni rappresentati relativamente dalle classi *GeoMarker* e *GeoPolygon*. Un marker è un'immagine di cui è possibile definire le seguenti proprietà:

Latitudine	Latitudine dove centrare l'immagine.
Longitudine	Longitudine dove centrare l'immagine.
ID	Identificativo del marker; da utilizzare nella gestione degli eventi di interfaccia legati al marker.
Icona	Immagine da mostrare. Può essere impostato ad uno dei valori della lista *IconeMarker* oppure ad un indirizzo internet di un immagine. Il valore predefinito è *PiccolaGoccia*.
Colore	Colore dell'immagine nel caso si usi un'immagine della lista valori IconeMarker. Il valore predefinito è *blue*.
Draggabile	Indica se il marker è trascinabile. Il valore predefinito è *false*.
Cliccabile	Indica se il marker è cliccabile. Il valore predefinito è *false*.
Tooltip	Testo da mostrare quando il mouse passa sopra al marker.

HTML	Testo HTML da mostrare nel fumetto che si apre quando l'utente clicca sul marker.

Un poligono è un insieme di punti di cui è possibile definire le seguenti proprietà:

ID	Identificativo del poligono; utile nel caso sia cliccabile.
ColoreBordi	Colore dei bordi in formato HTML. Il valore predefinito è *Red*.
ColoreRiempimento	Colore di riempimento in formato HTML. Il valore predefinito è *Red.*
SpessoreBordi	Spessore dei bordi in pixel. Il valore di default è 1 pixel.
OpacitàBordi	Livello di opacità dei bordi, tra 0, trasparente, e 1, pieno. Il valore predefinito è 0,5.
OpacitàRiempimento	Livello di opacità del riempimento, tra 0, trasparente, e 1, pieno. Il valore predefinito è 0,2.
Cliccabile	Indica se il poligono è cliccabile. Il valore predefinito è *false.*
Vertici	Insieme dei vertici del poligono; devono essere in sequenza.

Per aggiungere oggetti ad una mappa occorre prima pulirla con il metodo *ClearMap*, poi si possono aggiungere gli oggetti con il metodo *AddObjectToMap* ed infine si deve aggiornare la mappa chiamando *UpdateMap*. Se, ad esempio, si vuole visualizzare la mappa dell'Italia con un marker su Bologna occorre scrivere il seguente codice:

```
public void Videata.MostraBologna()
{
  // Creo il marker per mostrare dove è situata Bologna
  GeoMarker m = new()
  m.init()
  m.Icona = spillo
  m.Latitudine = 44,494
  m.Longitudine = 11,346
  m.Tooltip = "Bologna"
  //
  Mappa mapSF = Mappa.IDForm()
  GoogleMap map = mapSF.GetMapOptions()
  //
  // Aggiusto il livello di zoom
  map.ZoomLevel = 13
  //
  // Aggiungo il marker alla mappa
  mapSF.ClearMap()
  mapSF.AddObjectToMap(m)
  mapSF.UpdateMap()
}
```

4.5 Gestione degli eventi della mappa

Quando l'utente interagisce con la mappa alla videata in cui è ospitata la sotto-videata, vengono notificati degli eventi tramite il metodo SendMessage. Gestendo l'evento OnSendMessage della videata ospitante è possibile intercettare i seguenti eventi:

Evento	**Quando viene notificato?**
ClickPoint	Notificato quando l'utente clicca sulla mappa. Il parametro *Document* punta ad un oggetto *GeoPoint* che contiene le coordinate del punto cliccato.
MoveMarker	Notificato quando l'utente trascina un marker sulla mappa. Il parametro *Document* contiene un oggetto *GeoMarker* corrispondente al marker trascinato.
ClickMarker	Notificato quando l'utente clicca su un marker nella mappa. Il parametro *Document* contiene un oggetto *GeoMarker* corrispondente a quello cliccato.
ClickPolygon	Notificato quando l'utente clicca un poligono sulla mappa. Il parametro *Document* contiene un oggetto *GeoPolygon* corrispondente al poligono cliccato.
ChangeZoom	Notificato quando l'utente cambia il livello di zoom della mappa. Il nuovo livello di zoom viene aggiornato nella proprietà della mappa.
ChangeCenter	Notificato quando l'utente cambia il centro della mappa. Le coordinate del nuovo centro vengono aggiornate nelle proprietà della mappa.
ChangeMapType	Notificato quando l'utente cambia il tipo di mappa. Il nuovo tipo viene aggiornato nella proprietà della mappa.

Se si vuole intercettare, ad esempio, lo spostamento di un marker su una mappa occorre scrivere il seguente codice:

```
event Videata.OnSendMessage(
   string Message // Indica il nome del messaggio
   IDForm Sender  // Identifica la form che ha inviato il messaggio
   IDDocument Document // Optional document sent by the sender
   string Par1    // Opzionale. Contiene una espressione qualsiasi (il cu
   string Par2    // Opzionale. Contiene una espressione qualsiasi (il cu
   string Par3    // Opzionale. Contiene una espressione qualsiasi (il cu
   string Par4    // Opzionale. Contiene una espressione qualsiasi (il cu
)
{
  // Se i messaggi provengono dalla mappa
  if (Sender == Mappa.IDForm())
  {
    switch (Message)
    {
      case MoveMarker:
        GeoMarker m = GeoMarker.cast(Document)
        Applicazione.messageBox("E' stato spostato il marker " + m.ID)
      break
    }
  }
}
```

4.6 Geo-decodifica di indirizzi

La classe *GoogleMaps* mette a disposizione due metodi statici, *GeocodeAddress* e *ReverseGeocodeAddress*, che permettono di ottenere le coordinate geografiche di un indirizzo e viceversa.

In particolare, il metodo *GeocodeAddress* ha un parametro stringa in cui deve essere specificato l'indirizzo, e restituisce un oggetto GeoPoint che ne contiene le coordinate. All'inverso, il metodo *ReverseGeocodeAddress* richiede un oggetto *GeoPoint* in input e restituisce una stringa che contiene l'indirizzo. In questo caso è possibile indicare il tipo di indirizzo cercato tramite il parametro *AddressType* e la corrispondente lista valori.

Entrambi i metodi restituiscono il successo dell'operazione tramite i seguenti parametri di output:

1) *ResponseStatus*: viene valorizzato con il valore OK se la geo-decodifica ha avuto esito positivo. Se invece è stato superato il limite giornaliero di richieste si ottiene il valore OVER_QUERY_LIMIT. I possibili valori sono elencati nella lista valori *GeocodeStatus*.
2) *ResponseResult:* contiene il testo del documento xml di risposta ottenuto dal servizio di Google; può essere utile per ottenere maggiori dettagli.

Il seguente codice può essere usato se, ad esempio, si vogliono ottenere le coordinate geografiche dell'indirizzo "Via Indipendenza, Bologna, Italia":

```
public void Videata.Geodecodifica()
{
  string status = ""
  string result = ""
  string address = "Via Indipendenza, Bologna, Italy"
  GeoPoint coord = GoogleMap.GeocodeAddress(address, status, result)
  //
  switch (status)
  {
    case OK:
      Applicazione.messageBox("L'indirizzo si trova alle coordinate " +
          toString(coord.Latitudine) + ";" + toString(coord.Longitudine))
    break

    case ZERO_RESULTS:
    break

    case OVER_QUERY_LIMIT:
    break

    case REQUEST_DENIED:
    break

    case INVALID_REQUEST:
    break
  }
}
```

4.7 Calcolo di percorsi

Dalla versione 2.1 di GMAP, rilasciata con In.de 10.1, è disponibile anche la funzionalità di calcolo dei percorsi, che, a partire da una mappa contenuta in una videata, consente di ottenere le medesime informazioni della popolare versione web del servizio. Si noti che il calcolo avviene lato client in maniera asincrona, quindi non può essere paragonato alla chiamata di un web service.
Per ottenere il calcolo di un percorso occorre utilizzare la seguente procedura:

1) Creare un oggetto di tipo *GeoRoute*.
2) Valorizzare le proprietà *Partenza* e *Destinazione* usando le coordinate geografiche o l'indirizzo come stringa.
3) Ordinare alla mappa di calcolare il percorso tramite il metodo della mappa *CalculateRoute*, passando l'oggetto *GeoRoute* creato in precedenza.
4) Il calcolo avviene in modalità asincrona; quando è concluso la mappa notifica alla videata che la contiene l'evento OnSendMessage specificando come messaggio "CalcRoute" e passando nel parametro *Document* l'oggetto *GeoRoute* completo delle informazioni del percorso.

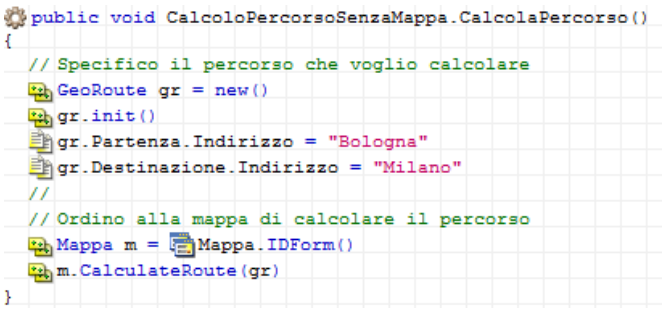
E' possibile configurare il tipo di percorso che si desidera calcolare tramite le seguenti proprietà:

EvitaPedaggi	Indica se il percorso deve evitare, dove possibile, strade con pedaggio. Il valore predefinito è *false*.
EvitaAutostrade	Indica se il percorso deve evitare, dove possibile, le autostrade. Il valore predefinito è *false*.
PiùCorto	Indica se deve essere calcolato il percorso più corto o quello più veloce. Il valore predefinito è *false*.
MezzoTrasporto	Mezzo di trasporto da utilizzare per calcolare il percorso; i possibili valori sono elencati nella lista valori *TravelModes*. Il valore predefinito è *DRIVING*.
MostraSuMappa	Indica se il percorso calcolato deve essere anche mostrato sulla mappa. Il valore predefinito è *true*.

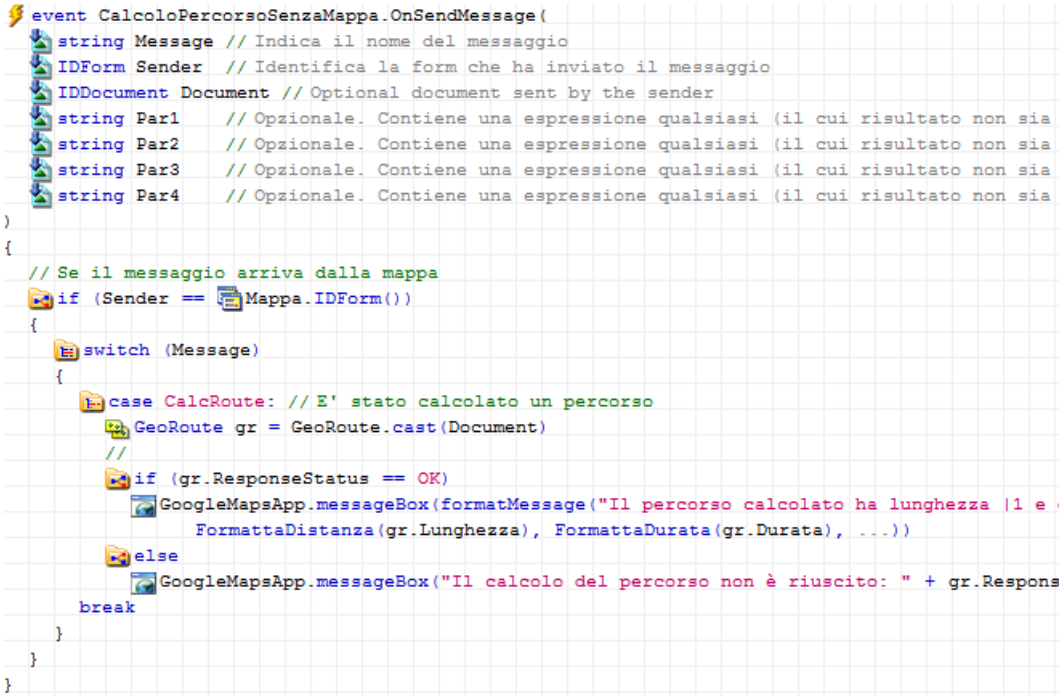
Le informazioni che vengono fornite nel calcolo sono le seguenti:

ResponseStatus	Esito del calcolo del percorso; il calcolo avvenuto con successo corrisponde al valore *OK*. I possibili valori sono elencati nella lista valori *RouteStatus*.
Lunghezza	Lunghezza totale del percorso espresso in metri.
Durata	Durata totale del viaggio espresso in secondi.
Steps	Lista dei passaggi da seguire per arrivare a destinazione. Ogni passaggio è corredato di *Lunghezza* e *Durata* (metri percorsi e secondi trascorsi nel passaggio) e di indicazioni in formato HTML.

Nell'immagine seguente viene mostrato un esempio di codice da utilizzare per il calcolo del percorso fra Bologna e Milano.



Quando il calcolo è completo possiamo reperire le informazioni tramite l'evento On-SendMessage, come mostrato nell'immagine seguente.



```
event CalcoloPercorsoSenzaMappa.OnSendMessage(
  string Message // Indica il nome del messaggio
  IDForm Sender  // Identifica la form che ha inviato il messaggio
  IDDocument Document // Optional document sent by the sender
  string Par1    // Opzionale. Contiene una espressione qualsiasi (il cui risultato non sia
  string Par2    // Opzionale. Contiene una espressione qualsiasi (il cui risultato non sia
  string Par3    // Opzionale. Contiene una espressione qualsiasi (il cui risultato non sia
  string Par4    // Opzionale. Contiene una espressione qualsiasi (il cui risultato non sia
)
{
  // Se il messaggio arriva dalla mappa
  if (Sender == Mappa.IDForm())
  {
    switch (Message)
    {
      case CalcRoute: // E' stato calcolato un percorso
        GeoRoute gr = GeoRoute.cast(Document)
        //
        if (gr.ResponseStatus == OK)
          GoogleMapsApp.messageBox(formatMessage("Il percorso calcolato ha lunghezza |1 e
              FormattaDistanza(gr.Lunghezza), FormattaDurata(gr.Durata), ...))
        else
          GoogleMapsApp.messageBox("Il calcolo del percorso non è riuscito: " + gr.Respons
      break
    }
  }
}
```

Se la mappa è mostrata a video, verrà visualizzato il percorso calcolato.

4.8 Licenza Premier

I servizi di visualizzazione mappe e di geo-codifica sono sottoposti ad alcuni limiti di utilizzo. In particolare, ad oggi è possibile effettuare un massimo di 2.500 richieste di geo-codifica giornaliere ed è importante rispettare questi vincoli temporali, pena l'interruzione temporanea o definitiva del servizio.

Acquistando una licenza Premier per i servizi di Google si può godere di maggiori funzionalità; fra cui:

1) capacità di geo-codifica avanzate, con volumi e velocità più elevati;
2) trasmissione tramite protocollo HTTPS per integrare le mappe in siti protetti;
3) tracciamento dei rapporti di utilizzo su più domini.

Per godere i vantaggi della licenza Premium basta valorizzare i campi *ClientID* e *PrivateKey* della tabella *Parametri*. Tali informazioni verranno comunicate ai servizi Google sia per la visualizzazione delle mappe che per le funzionalità di geo-decodifica.

Capitolo 5

RTC Designer

5.1 Il Componente RTC Designer

Il componente *RTC Designer*, disponibile dalla versione 10.1, permette di aggiungere alle proprie applicazioni le stesse funzionalità di personalizzazione presenti nel modulo di installazione *IDManager*. Le videate contenute sono elencate sotto; per una descrizione del loro funzionamento vedere anche il paragrafo *13.5 RTC Designer* della guida all'uso.

Designer	Permette di configurare i vari oggetti dell'applicazione in base alle proprietà RTCLanguage, RTCUserID e RTCGroupID. La configurazione dei pannelli, delle pagine mastro e dei report avviene anche tramite una videata di anteprima.
GestioneTraduzioni	Permette di tradurre tutte le frasi presenti nell'applicazione; la traduzione può anche essere fatta automaticamente sfruttando i servizi di traduzione di Google e Microsoft.
ConfigurazioneRuoli	Permette di definire le limitazioni dei ruoli dell'applicazione, di creare nuovi ruoli, di cambiarne la gerarchia e di avere un'anteprima di ciò che potrà vedere e fare un utente a seguito dell'abilitazione di una sequenza di ruoli.

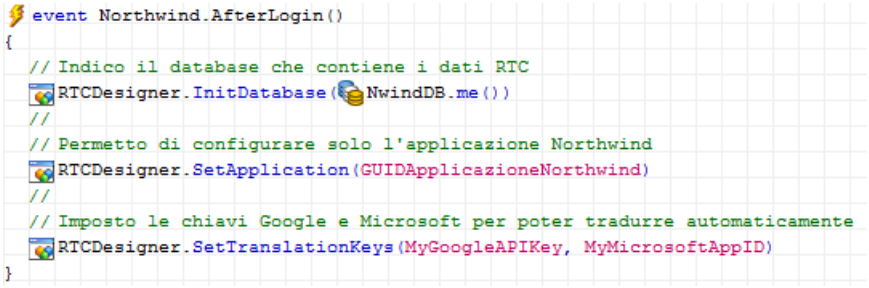
5.2 Inizializzazione del componente

Dopo aver importato il componente nel progetto occorre effettuare alcune operazioni di inizializzazione. In primo luogo occorre indicare il database che contiene i dati RTC usando la funzione *InitDatabase*. Se il database contiene i dati RTC di più applicazioni si può limitare la configurazione ad una specifica applicazione grazie al metodo SetApplication a cui va passato il GUID dell'applicazione da configurare.

Se si vogliono utilizzare i servizi di traduzione di Google o Microsoft, occorre chiamare il metodo *SetTranslationKeys* passandogli le proprie chiavi di identificazione

ottenibili alle pagine http://code.google.com/apis/console e http://www.bing.com/developers/appids.aspx. Occorre tenere presente che il servizio di Google consente la traduzione di 100.000 caratteri/giorno, mentre quello di Microsoft impone dei limiti al numero di richieste orarie e giornaliere in base al carico del servizio.

Nell'immagine seguente viene mostrato come inizializzare RTC Designer.



Esempio di inizializzazione del componente.

Il componente mette a disposizione un menù contestuale che permette di accedere alle videate. Questo menù può essere utilizzato dove lo si ritiene più opportuno. In alternativa le videate sono apribili tramite le procedure *OpenRTCDesigner*, *OpenRoleConfigurator* e *OpenTranslator*.

Capitolo 6

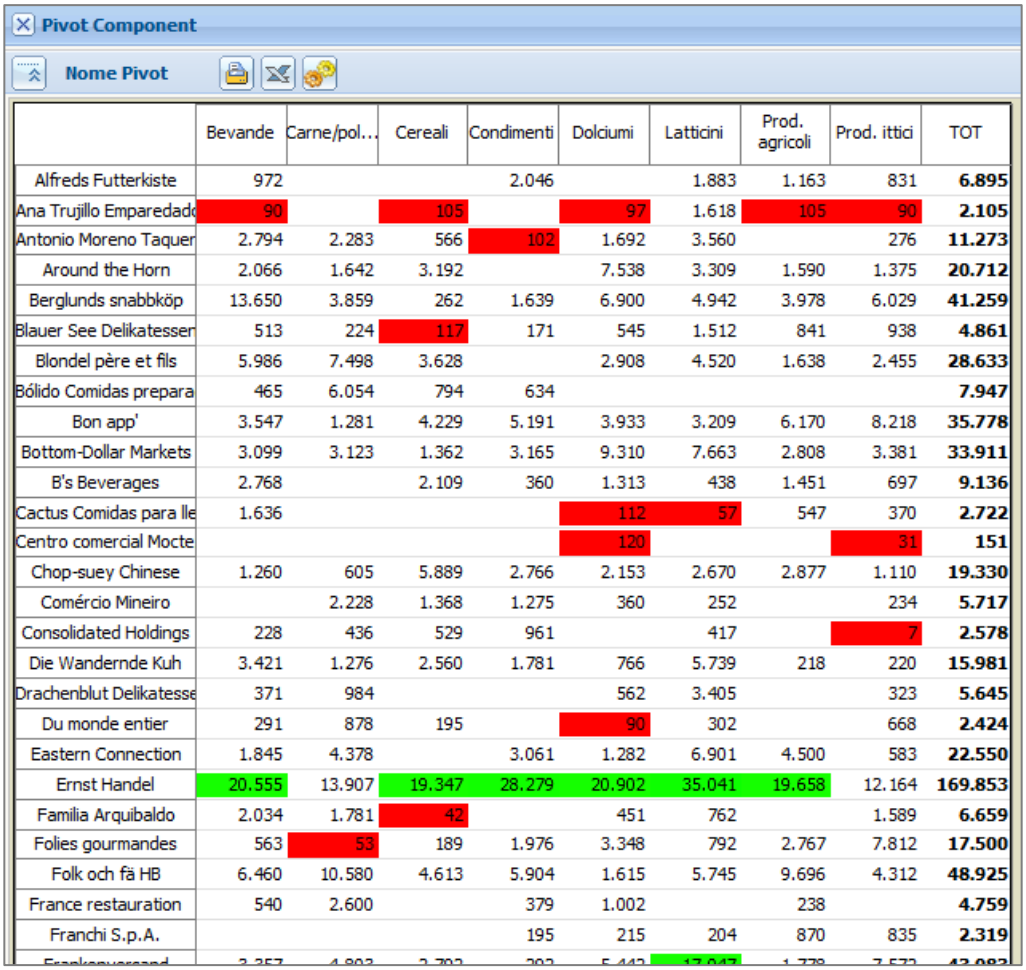
Pivot

6.1 Il Componente Pivot

Il componente Pivot è uno strumento applicativo aggiuntivo messo a disposizione del programmatore che permette la visualizzazione e la configurazione di tabelle pivot con poche righe di codice. Le funzionalità principali di questo componente sono descritte di seguito:

1) Fino a tre dimensioni (livelli di raggruppamento) verticali collassabili.
2) Fino a tre dimensioni orizzontali collassabili.
3) Fino a tre filtri autoalimentati e generati sia da codice che tramite drag&drop da ciascun raggruppamento.
4) Fino a cinque misure (campi valore aggregati) con le operazioni più comuni: somma, media, minimo, massimo, conteggio.
5) Possibilità di colorare il testo e lo sfondo dei dati delle misure mediante la definizione di uno o più range, anche in forma percentuale.
6) Possibilità di visualizzare singolarmente sia i totali di raggruppamento, che di foglio.
7) Videata di configurazione per l'utente finale a runtime della tabella pivot.
8) Esportazione e stampa in CSV e PDF.

L'immagine seguente mostra un esempio di tabella pivot realizzabile con questo componente e qualche riga di codice.



	Bevande	Carne/pol…	Cereali	Condimenti	Dolciumi	Latticini	Prod. agricoli	Prod. ittici	TOT
Alfreds Futterkiste	972			2.046		1.883	1.163	831	**6.895**
Ana Trujillo Emparedado	90		105		97	1.618	105	90	**2.105**
Antonio Moreno Taquer	2.794	2.283	566	102	1.692	3.560		276	**11.273**
Around the Horn	2.066	1.642	3.192		7.538	3.309	1.590	1.375	**20.712**
Berglunds snabbköp	13.650	3.859	262	1.639	6.900	4.942	3.978	6.029	**41.259**
Blauer See Delikatesser	513	224	117	171	545	1.512	841	938	**4.861**
Blondel père et fils	5.986	7.498	3.628		2.908	4.520	1.638	2.455	**28.633**
Bólido Comidas prepara	465	6.054	794	634					**7.947**
Bon app'	3.547	1.281	4.229	5.191	3.933	3.209	6.170	8.218	**35.778**
Bottom-Dollar Markets	3.099	3.123	1.362	3.165	9.310	7.663	2.808	3.381	**33.911**
B's Beverages	2.768		2.109	360	1.313	438	1.451	697	**9.136**
Cactus Comidas para lle	1.636				112	57	547	370	**2.722**
Centro comercial Mocte					120			31	**151**
Chop-suey Chinese	1.260	605	5.889	2.766	2.153	2.670	2.877	1.110	**19.330**
Comércio Mineiro		2.228	1.368	1.275	360	252		234	**5.717**
Consolidated Holdings	228	436	529	961		417		7	**2.578**
Die Wandernde Kuh	3.421	1.276	2.560	1.781	766	5.739	218	220	**15.981**
Drachenblut Delikatesse	371	984			562	3.405		323	**5.645**
Du monde entier	291	878	195		90	302		668	**2.424**
Eastern Connection	1.845	4.378		3.061	1.282	6.901	4.500	583	**22.550**
Ernst Handel	20.555	13.907	19.347	28.279	20.902	35.041	19.658	12.164	**169.853**
Familia Arquibaldo	2.034	1.781	42		451	762		1.589	**6.659**
Folies gourmandes	563	53	189	1.976	3.348	792	2.767	7.812	**17.500**
Folk och fä HB	6.460	10.580	4.613	5.904	1.615	5.745	9.696	4.312	**48.925**
France restauration	540	2.600		379	1.002		238		**4.759**
Franchi S.p.A.				195	215	204	870	835	**2.319**

Queste funzionalità possono essere ottenute tramite l'insieme dei metodi esposti dal componente, che verranno esaminati in dettaglio nel paragrafo seguente.

6.2 Metodi a disposizione del programmatore

ClearConfiguration	Inizializza la configurazione della tabella.
SetPivotData	Imposta i dati da visualizzare nella tabelle passando un oggetto Recordset.

SetPivotName	Definisce il nome della pivot e lo mostrerà come titolo del pannello
SetCellSize	Definisce larghezza e altezza della cella dei campi valore.
AddVerticalDimension	Aggiunge una dimensione verticale. Ha come parametro obbligatorio il nome della colonna del recordset che dà origine alla dimensione e due parametri opzionali che permettono di richiedere i totali e di specificare se inizialmente il livello deve essere collassato.
AddHorizontalDimension	Aggiunge una dimensione orizzontale. Ha come parametro obbligatorio il nome della colonna del recordset che dà origine alla dimensione e due parametri opzionali che permettono di richiedere i totali e di specificare se inizialmente il livello deve essere collassato.
AddCellField	Definisce una misura (campo valore). Ha come parametri obbligatori il nome della colonna del recordset che li contiene e l'operazione di aggregazione, oltre a quattro parametri opzionali che ne definiscono le caratteristiche grafiche (TextColor, Mask, Enabled, Range).
AddColumnFilter	Aggiunge un filtro specificando il nome della colonna del Recordset da condizionare, e un parametro opzionale per definire il watermark.
ShowPageTotals	Permette la configurazione della visualizzazione dei totali di foglio orizzontali e verticali.
AddRange	Definisce un range per la colorazione dello sfondo delle celle.
UpdateTable	Aggiorna, ricalcola e ridisegna la tabella pivot, in funzione della configurazione e dei dati attuali.
GetXML	Restituisce una stringa in formato XML al cui interno è presente la configurazione della pivot.
LoadFromXML	Carica la configurazione della pivot da una stringa XML passata come parametro.

6.3 Creazione di una tabella Pivot

Vediamo ora come utilizzare i metodi a disposizione per realizzare una tabella pivot come quella visualizzata nell'immagine di presentazione del componente. Dopo aver

scaricato dalla component gallery il componente Pivot, è necessario importarlo nel proprio progetto con il comando *Importa Componente* nel menù contestuale del progetto.

A questo punto è sufficiente creare una procedura che estrae i dati dal database e configura la pivot. Nell'immagine seguente viene mostrata l'istruzione che esegue l'estrazione dei dati che devono poi essere passati tramite il metodo *SetPivotData*.

```
public void NuovaApplicazioneWeb.MostraPivot()
{
  Recordset r = new() //
  select into recordset (r)
    Prodotti.NomeProdotto as PRODOTTO
    Categorie.NomeCategoria as CATEGORIA
    f(x) round(month(Ordini.DataOrdine) / 3 + 0,49, 0) as QUARTER
    f(x) month(Ordini.DataOrdine) as MESI
    f(x) year(Ordini.DataOrdine) as ANNO
    f(x) round(DettagliOrdini.Sconto * 100, 0) as SCONTO
    f(x) round((DettagliOrdini.PrezzoUnitario * DettagliOrdini.Quantita) / 1000,
      0) as VENDUTO
    DettagliOrdini.Quantita as QUANTITA
    f(x) Impiegati.Cognome + " " + Impiegati.Nome as NOMEIMPIEGATO
    Ordini.Idcliente as CLIENTE
    Clienti.NomeSocieta as NOMECLIENTE
  from
    Prodotti  // master table
    Categorie // joined with Prodotti using key CategoriaProdotti
    DettagliOrdini // joined with Prodotti using key ProdottiDettagli_ordi...
    Ordini    // joined with Dettagli Ordini using key OrdiniDettagli_ordini
    Clienti   // joined with Ordini using key ClientiOrdini
    Impiegati // joined with Ordini using key ImpiegatiOrdini
```

La configurazione della pivot avviene tramite le seguenti righe di codice:

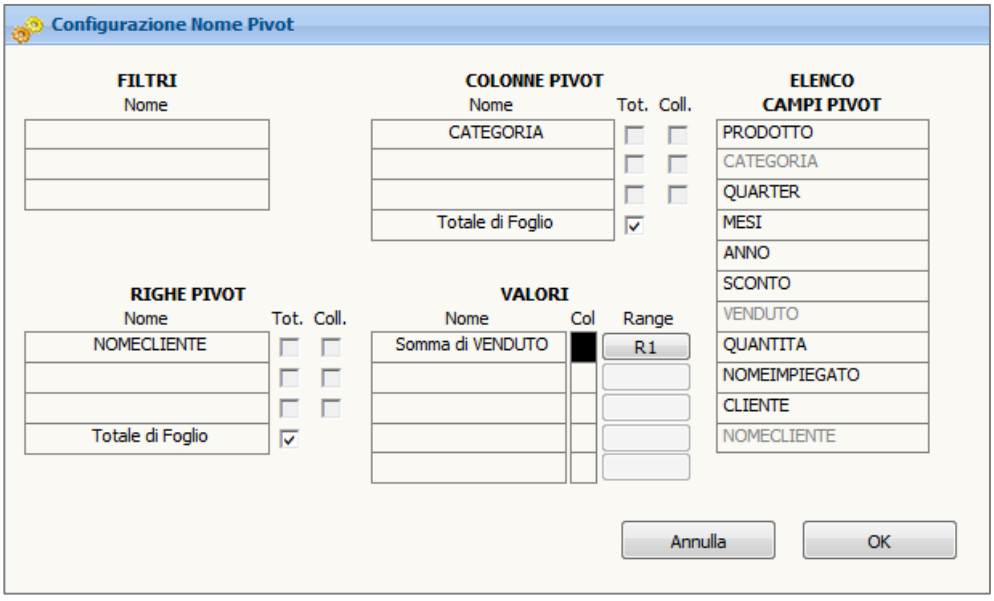
```
// Inizializzo la tabella Pivot
PivotComponent.ClearConfiguration(...)
//
// Memorizzo il RecordSet in una variabile locale
PivotComponent.SetPivotData(r)
//
// Imposto il nome della Pivot che verrà visualizzata nel titolo del pannello
PivotComponent.SetPivotName("Nome Pivot")
//
// Aggiungo un raggruppamento orizzontale per "NOMECLIENTE", visualizzando i
// totali e collassandolo
PivotComponent.AddHorizontalDimension("NOMECLIENTE", true, true)
//
// Aggiungo un raggruppamento verticale per "CATEGORIA", visualizzando i
// totali e collassandolo
PivotComponent.AddVerticalDimension("CATEGORIA", true, true)
```

```
// Aggiungo un campo valore per il "VENDUTO" di cui farò la somma e a cui
// applicherò un range di colori di nome "R1"
PivotComponent.AddCellField("VENDUTO", Somma, ..., "R1")
//
// Imposto il range "R1" in due possibili intervalli
PivotComponent.AddRange("R1", RGBColor(255, 0, 0, ...), RGBColor(255, 0, 0
    , ...), 0, 20, true)
PivotComponent.AddRange("R1", RGBColor(0, 255, 0, ...), RGBColor(0, 255, 0
    , ...), 80, 100, true)
//
// Visualizzo i totali di pagina, sia per i raggruppamenti orizzontali che
// per quelli verticali
PivotComponent.ShowPageTotals(true, true)
//
// Calcolo il necessario e disegno la tabella Pivot a video
PivotComponent.UpdateTable()
```

Per renderla maggiormente dettagliata, è possibile chiamare le ulteriori funzioni disponibili e descritte nel capitolo precedente. Dopo aver compilato l'applicazione, è sufficiente eseguire la procedura per verificare l'effettivo funzionamento della pivot.

6.4 Configurazione di una tabella Pivot

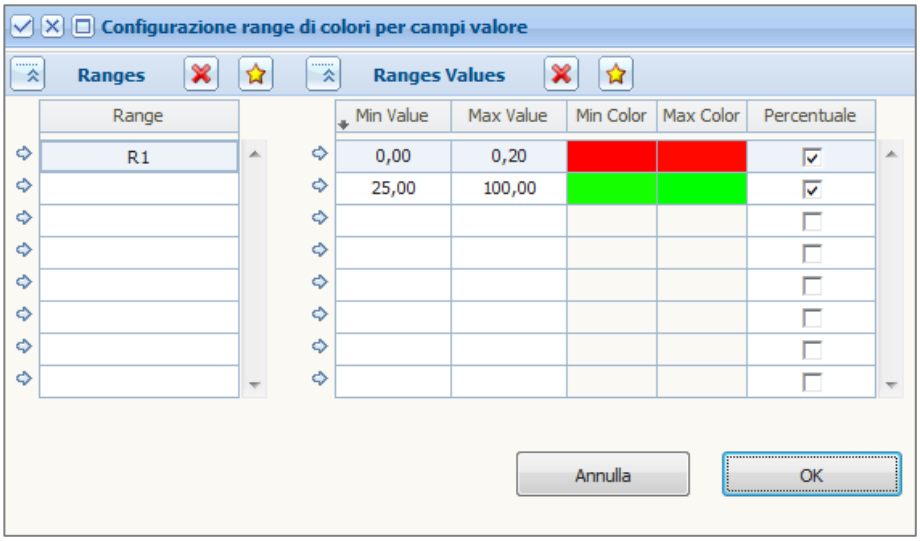
Nella barra del titolo della pivot è presente un pulsante di configurazione  che permette all'utente finale di riconfigurarla a piacimento. Premendolo si apre la seguente videata.



Esempio di videata di configurazione della pivot

Tramite drag&drop è possibile trascinare i dati disponibili nel recordset (elenco campi pivot) utilizzandoli come dimensione orizzontale, verticale o come misura. I dati già usati vengono colorati in grigio. E' possibile scegliere se aggiungere le totalizzazioni e se le dimensioni devono essere inizialmente collassate. Si consiglia di mantenere l'impostazione di default per minimizzare i tempi di visualizzazione della pivot.

Cliccando sul nome di una misura (valori) è possibile scegliere l'operazione di aggregazione; inoltre si può cambiare il colore del testo e specificare un range per lo sfondo. La configurazione dei range avviene tramite la seguente videata che appare in popup.



Esempio di videata di configurazione dei range per lo sfondo delle celle

In questo esempio, il range R1 permette di colorare in rosso tutti i valori rientranti nell'intervallo 0-0.2% rispetto al valore minimo della misura e in verde quelli tra il 25 al 100 della stessa. In questo modo è possibile evidenziare i dati posizionati agli estremi del campo di misurazione.

Per salvare su file o su database la configurazione della pivot modificata dall'utente, è possibile utilizzare il metodo *GetXML()* che restituisce una stringa XML che la rappresenta completamente. Per caricare una configurazione precedentemente salvata deve essere chiamata la funzione *LoadFromXML()*.

.

Capitolo 7

Componente IDCloud

7.1 Introduzione

Il componente *IDCloud* permette di integrare vari servizi cloud all'interno delle applicazioni realizzate con Instant Developer. I servizi a disposizione sono i seguenti:

- Integrazione con Paypal per permettere di eseguire pagamenti direttamente dall'applicazione.
- Login tramite Facebook e accesso ai contenuti pubblicati su Facebook.
- Login tramite Google.
- Integrazione con Google Calendar per la gestione dell'agenda.
- Integrazione con Gmail per la lettura e l'invio di email.
- Integrazione con Dropbox, Google Drive e SkyDrive.

Per includere il componente nelle proprie applicazioni è sufficiente importare il file *IDCloud.idz* nel progetto e poi seguire le istruzioni descritte nei paragrafi successivi. Nell'application gallery di Instant Developer è presente il progetto di esempio IDCloud che mostra come utilizzare il componente nei casi più comuni.

7.2 Servizio Paypal

Il servizio Paypal permette di accettare pagamenti con Paypal all'interno delle applicazioni realizzate con Instant Developer. Le funzionalità del servizio sono le seguenti:

- ricezione pagamenti con Paypal;
- ricerca transazioni;
- saldo del conto.

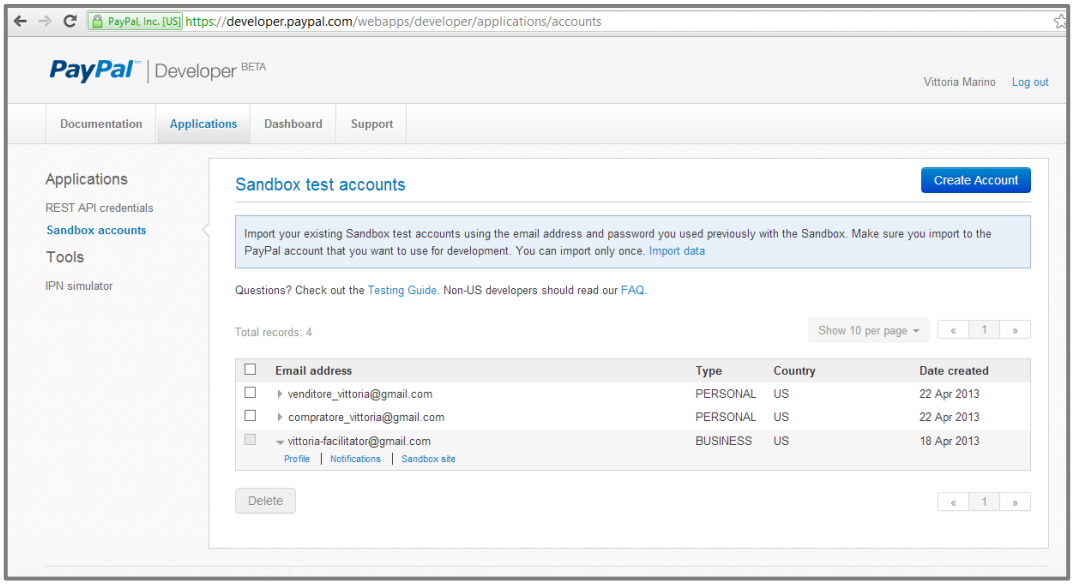
Per poterlo utilizzare è necessario essere in possesso delle credenziali. Si possono richiedere due tipi di credenziali: credenziali di test e credenziali di produzione.

7.2.1 Richiesta credenziali di test

Paypal mette a disposizione Paypal Sandbox, un ambiente di test con le stesse funzionalità del vero sito Paypal che può essere utilizzato per testare le applicazioni con account fittizi senza coinvolgere utenti e conti reali.

Per poter utilizzare il servizio è necessario registrarsi come sviluppatore Paypal all'indirizzo https://developer.paypal.com/

La gestione degli account di test può essere fatta dal sito sviluppatori di Paypal cliccando su *Applications* nella barra di navigazione in alto e poi su *Sandbox accounts* nel menu laterale di sinistra.



Pagina relativa agli account di test nel sito sviluppatori di Paypal

All'atto della registrazione viene automaticamente creato un account di test di tipo business al quale verranno associate le credenziali (*Username*, *Password* e *Signature*) per l'utilizzo di Paypal in modalità sandbox. Questo account rappresenterà il venditore di prova.

Per sapere quali sono le credenziali basta cliccare sull'indirizzo email dell'account Business, poi su *Profile* e infine, nella finestra che si aprirà, su *API credentials*. La creazione di nuovi account di test può essere fatta cliccando sul pulsante *Create Account* in alto a destra.

7.2.2 Richiesta credenziali di produzione e pubblicazione dell'applicazione

Quando l'applicazione sarà pronta, per poterla mettere in produzione è necessario ottenere delle nuove credenziali e registrare l'applicazione su Paypal. Per ottenere queste credenziali è necessario avere un account Paypal Premier o Business verificato ed eseguire le seguenti operazioni:

- Login nel sito di Paypal con l'account associato all'applicazione.
- Navigare fino alla pagina *My Selling Tools* selezionando *My Account* > *Profile* > *My Selling Tools*.
- Cliccare su *Selling Online* > *API Access* > *Update* per visualizzare la pagina relativa alle API.
- Sotto *Option 2*, cliccare *View API Signature* per arrivare alla pagina relativa alle credenziali. Se sono già state richieste delle credenziali, saranno visualizzate in questa pagina.
- Per creare nuove credenziali, selezionare *Request API Signature* e cliccare su *Agree and Submit*.

Per registrare l'applicazione su Paypal:

- Assicurarsi che l'applicazione rispetti le linee guida di Paypal.
- Effettuate il *Login* nel sito Paypal Developer Network usando l'account associato all'applicazione.
- Cliccare su *My Account* nella barra di navigazione in alto e poi scegliere *My Application* > *New App* dal menu laterale sulla sinistra per accedere alla pagina per la registrazione dell'applicazione.
- Riempire il modulo facendo attenzione a fornire lo stesso indirizzo email usato per richiedere le credenziali e indicando come tipo di applicazione *Merchant APIs*.

Dopo la registrazione l'applicazione è immediatamente utilizzabile anche se sarà soggetta, nelle successive 24-72 ore, a un processo di verifica; si sarà contattati da Paypal solo nel caso in cui ci siano dei problemi. È bene perciò controllare il sito Paypal Developer Network durante il periodo di verifica dell'applicazione.

7.2.3 Accettare pagamenti

Vediamo adesso come utilizzare gli oggetti messi a disposizione nel componente IDCloud per gestire il proprio account Paypal, ed in particolare per accettare pagamenti da parte di un utente di una propria applicazione.

Per prima cosa è necessario creare un oggetto di tipo *PayPalServer* e impostarne le proprietà. Quelle disponibili sono le seguenti:

Username	Username assegnato al momento della registrazione sul sito sviluppatori nel caso di account sandbox o al momento della richiesta delle credenziali nel caso dell'account di produzione.
Signature	Signature assegnata al momento della registrazione sul sito sviluppatori nel caso di account sandbox o al momento della richiesta delle credenziali nel caso dell'account di produzione.
Password	Password assegnata al momento della registrazione sul sito sviluppatori nel caso di account sandbox o al momento della richiesta delle credenziali nel caso dell'account di produzione.
DebugMode	Indica se l'applicazione userà l'ambiente Sandbox o il server di produzione PayPal
Idform	Id della form che riceverà i messaggi inviati dal componente.

Occorre quindi creare un oggetto di tipo *Cart* e impostarne le proprietà. Quelle disponibili sono le seguenti:

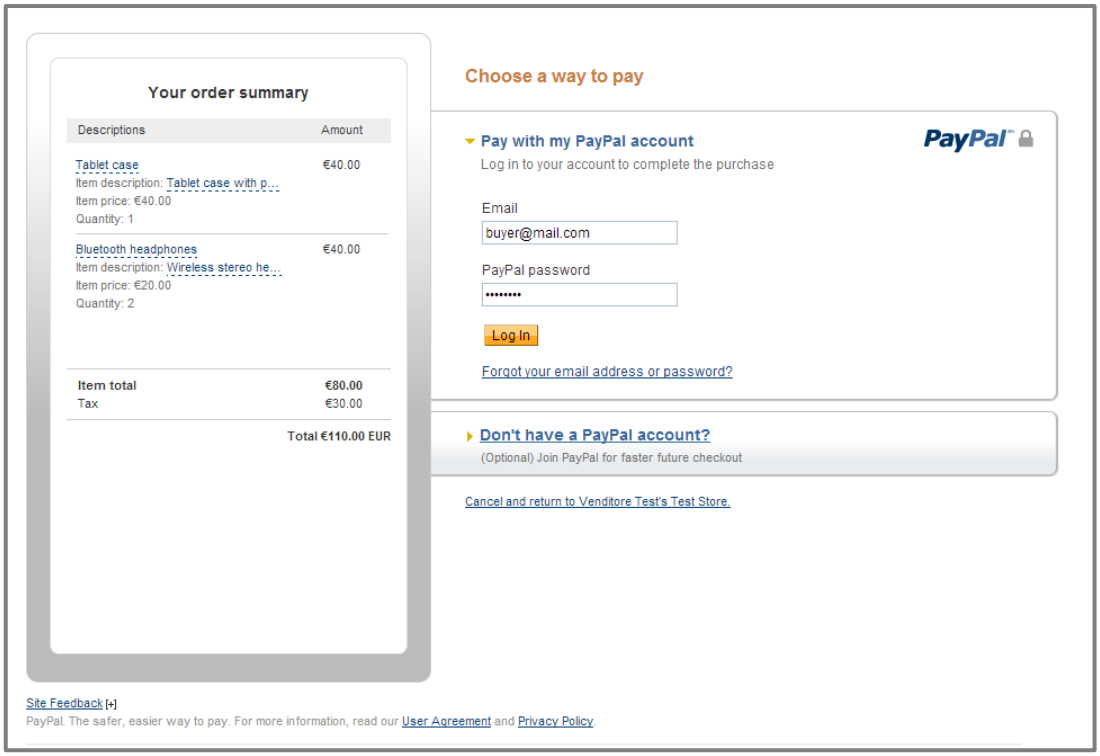
Amount	(obbligatorio) Somma che il compratore dovrà pagare. Devono essere incluse in questo valore anche gli eventuali costi di spedizione e le tasse.
Currency Code	(obbligatorio) Codice di tre caratteri che rappresenta la valuta. Per la lista dei codici vedi: https://developer.paypal.com/webapps/developer/docs/classic/api/currency_codes/
Tax	Somma delle tasse dovute per tutti gli articoli compresi nella transazione.
Description	Descrizione dei beni oggetto della transazione.
Email	Indirizzo e-mail del compratore.
Name	Nome del compratore.
Street	Nome della via dell'indirizzo del compratore.
City	Città del compratore.
State	Stato o provincia del compratore.
Zip	Codice postale del compratore.
Country Code	Codice di due caratteri che rappresenta la nazione del compratore. Per la lista dei codici vedi: https://developer.paypal.com/webapps/developer/docs/classic/api/country_codes/
Billing Type	Nel caso in cui il pagamento sia un pagamento ricorrente o una sottoscrizione, il valore di questa proprietà deve essere *RecurringPayment*. In tutti gli altri casi deve essere lasciata vuota.

Billing Agreement Description	Descrizione dei beni o servizi associati al tipo di pagamento impostato nel parametro Billing Type.https://developer.paypal.com/webapps/developer/docs/classic/api/merchant/SetExpressCheckout_API_Operation_NVP/
Checkout Items	Collection di oggetti di tipo Checkout Item.

Per aggiungere al carrello gli articoli occorre creare un oggetto di tipo *CheckoutItem* per ogni articolo e aggiungerlo alla collection *CheckoutItems* dell'oggetto *Cart*. Le proprietà dell'oggetto *CheckoutItem* sono le seguenti:

Name	Nome dell'articolo.
Description	Descrizione dell'articolo.
Amount	Prezzo unitario dell'articolo.
Quantity	Quantità acquistata.
Tax	Tasse relative ad un articolo.

Per ricevere il pagamento occorre usare il metodo *DoCheckout* che reindirizza il compratore alla pagina di PayPal.



Pagina di PayPal visualizzata dal compratore

Il compratore compirà le seguenti azioni:

- si autentica con il proprio indirizzo email e la password;
- rivede le informazioni di spedizione e il riepilogo dell'ordine. I dettagli dell'ordine saranno visibili solo se nell'oggetto *CheckoutItem* erano state valorizzate le proprietà relative ai singoli articoli (*Item Name*, *Item Description*, *Item Amount*, *Item Quantity*, *Item Tax*);
- se il compratore conferma l'ordine viene fatta una verifica di tutte le informazioni relative ad esso e si riceverà il pagamento;
- se il compratore non conferma l'ordine verrà mandato un messaggio *Checkout Error* all'Idform e verrà valorizzata la variabile globale *LastErrorMessage*.

In generale, qualsiasi errore si dovesse verificare in questo flusso di operazioni verrà notificato con un messaggio *Checkout Error* all'Idform e verrà valorizzata la variabile globale *LastErrorMessage*. Se invece l'operazione va a buon fine, verrà inviato un messaggio *Checkout Success* all'Idform. Nell'evento *OnSendMessage* si avrà a disposizione l'oggetto di tipo *PaypalServer* e, nel parametro *par1*, una stringa che spiega il motivo del fallimento dell'operazione.

Se, ad esempio, si vuole permettere al compratore John Smith di procedere al pagamento del suo ordine di una cover per un tablet e due auricolari Bluetooth occorre scrivere il seguente codice associandolo, per esempio, a un pulsante *Pagamento PayPal*:

```
public void Form.Checkout()
{
  // create a Paypal Server object
  PaypalServer ps = new()
  ps.Username = "vendit_1360837597_biz_api1.gmail.com"
  ps.Signature = "A5zZOi.7c9ngcAOVmfd7hjJbH9I4AZi-XT1DjebHfKlSqSCh2xbzjkV3"
  ps.Password = "1360837621"
  ps.DebugMode = true
  //
  // create a Cart object
  Cart c = new()
  c.init()
  c.Amount = 110,00
  c.CurrencyCode = "EUR"
  c.Tax = 30,00
  c.ItemsAmount = 80,00
  c.Email = "buyer@mail.com"
  c.Name = "John Smith"
  c.Street = "1 Main St"
  c.City = "San Jose"
  c.Zip = "95131"
  c.CountryCode = "US"
  c.State = "CA"
```

```
// create Checkout Item objects
CheckoutItem tabletCase = new()
tabletCase.init()
tabletCase.Amount = 40,00
tabletCase.Name = "Tablet case"
tabletCase.Description = "Tablet case with pocket"
tabletCase.Tax = 10,00
tabletCase.Quantity = 1
c.CheckoutItems.add(tabletCase)
//
CheckoutItem headphones = new()
headphones.init()
headphones.Amount = 20,00
headphones.Name = "Bluetooth headphones"
headphones.Description = "Wireless stereo headphones (white and orange)"
headphones.Tax = 10,00
headphones.Quantity = 2
c.CheckoutItems.add(headphones)
//
c.saveToDB(...)
//
// handle the payment
ps.DoCheckout(c)
}
```

7.2.4 Lista movimenti

Dopo aver creato un oggetto di tipo *PayPal Server*, occorre creare un oggetto di tipo *Movement Item* e impostarne le proprietà. Ogni proprietà valorizzata verrà utilizzata come filtro per la ricerca, se non ne viene impostata nessuna verranno cercate tutte le transazioni dell'anno corrente. Le proprietà disponibili sono le seguenti:

ID	ID della transazione.
Amount	Ammontare totale della transazione.
Currency Code	Codice di tre caratteri che rappresenta la valuta.
Email	Indirizzo e-mail del compratore.
Status	Stato della transazione. Può assumere uno dei seguenti valori: Pending, Processing, Success, Denied, Reversed.
Start Date	La data dalla quale iniziare la ricerca.
End Date	La data fino alla quale restituire risultati della ricerca.
Class	Classificazione della transazione. Può assumere uno dei seguenti valori: All, Sent, Received, MassPay, MoneyRequest, FundsAdded, FundsWithdrawn, Referral, Fee, Subscription, Dividend, Billpay, Refund, CurrencyConversions, BalanceTransfer, Reversal, Shipping, BalanceAffecting, ECheck.

Per effettuare la ricerca è necessario passare l'oggetto *MovementItem* al metodo *SearchTransaction* della classe *PayPalServer* che popolerà la collection *Movements* dell'oggetto *PayPalServer*. Se, ad esempio, si vogliono cercare tutte le transazioni del 2012 occorre scrivere il seguente codice:

```
public void Form.ButtonSearchTransactions()
{
  // create a Paypal Server object
  PaypalServer ps = new()
  ps.Username = "vendit_6984552361_biz_api1.gmail.com"
  ps.Signature = "A5zZOi.7c9ngcAOVmfd7hjJbH9I4AZi-XT1DjebHfKlSqSCh2xbzjkV3"
  ps.Password = "652115677"
  ps.DebugMode = true
  //
  // create a Movement Item object
  MovementItem mi = new()
  mi.StartDate = toDate(2012, 1, 1)
  mi.EndDate = toDate(2012, 12, 31)
  //
  // search transactions
  ps.SearchTransaction(mi)
}
```

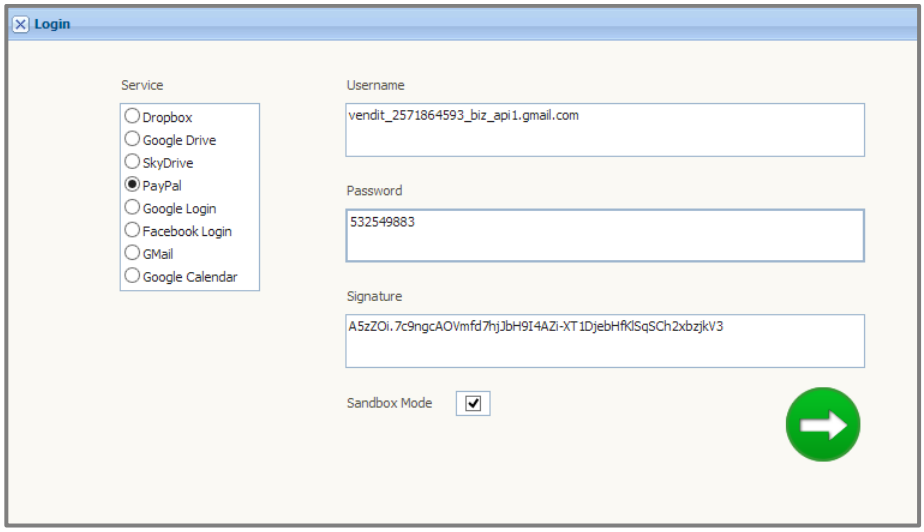
7.2.5 Ottenere il saldo del conto

Per ottenere il saldo del conto è necessario usare il metodo *GetBalance* che ha una parametro float in cui verrà restituito il saldo del conto e un parametro di tipo stringa in cui verrà restituita la valuta. Ecco un esempio di codice:

```
public void Paypal.GetBalance()
{
  // get paypal server
  PaypalServer ps = new()
  //
  ps.Username = "vendit_6984552361_biz_api1.gmail.com"
  ps.Password = "652115677"
  ps.Signature = "A5zZOi.7c9ngcAOVmfd7hjJbH9I4AZi-XT1DjebHfKlSqSCh2xbzjkV3"
  ps.Idform = this.IDForm()
  //
  if (Login.Data.PaypalSandboxMode == sandbox)
    ps.DebugMode = true
  else  //
    ps.DebugMode = false
  //
  // get balance
  float balance = 0
  string currency = ""
  ps.GetBalance(balance, currency)
  MovementsList.LabelBalance.text = toString(balance) + " " + currency
}
```

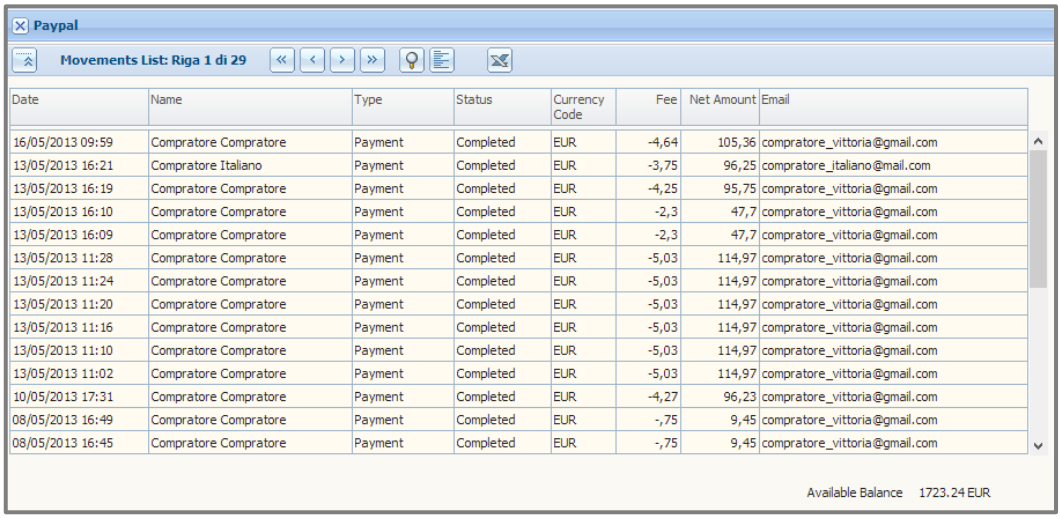
7.2.6 Test del servizio Paypal

Per provare il servizio Paypal tramite l'applicazione di esempio IDCloud occorre selezionare PayPal nella schermata di login dell'applicazione di esempio e poi scrivere le credenziali nei rispettivi campi, selezionando la check box Sandbox Mode se si stanno usando credenziali di test.



Videata di login dell'applicazione di esempio

Cliccando sul bottone con la freccia verrà visualizzata la videata con la lista dei movimenti relativi al conto associato alle credenziali inserite.



Date	Name	Type	Status	Currency Code	Fee	Net Amount	Email
16/05/2013 09:59	Compratore Compratore	Payment	Completed	EUR	-4,64	105,36	compratore_vittoria@gmail.com
13/05/2013 16:21	Compratore Italiano	Payment	Completed	EUR	-3,75	96,25	compratore_italiano@mail.com
13/05/2013 16:19	Compratore Compratore	Payment	Completed	EUR	-4,25	95,75	compratore_vittoria@gmail.com
13/05/2013 16:10	Compratore Compratore	Payment	Completed	EUR	-2,3	47,7	compratore_vittoria@gmail.com
13/05/2013 16:09	Compratore Compratore	Payment	Completed	EUR	-2,3	47,7	compratore_vittoria@gmail.com
13/05/2013 11:28	Compratore Compratore	Payment	Completed	EUR	-5,03	114,97	compratore_vittoria@gmail.com
13/05/2013 11:24	Compratore Compratore	Payment	Completed	EUR	-5,03	114,97	compratore_vittoria@gmail.com
13/05/2013 11:20	Compratore Compratore	Payment	Completed	EUR	-5,03	114,97	compratore_vittoria@gmail.com
13/05/2013 11:16	Compratore Compratore	Payment	Completed	EUR	-5,03	114,97	compratore_vittoria@gmail.com
13/05/2013 11:10	Compratore Compratore	Payment	Completed	EUR	-5,03	114,97	compratore_vittoria@gmail.com
13/05/2013 11:02	Compratore Compratore	Payment	Completed	EUR	-5,03	114,97	compratore_vittoria@gmail.com
10/05/2013 17:31	Compratore Compratore	Payment	Completed	EUR	-4,27	96,23	compratore_vittoria@gmail.com
08/05/2013 16:49	Compratore Compratore	Payment	Completed	EUR	-,75	9,45	compratore_vittoria@gmail.com
08/05/2013 16:45	Compratore Compratore	Payment	Completed	EUR	-,75	9,45	compratore_vittoria@gmail.com

Cliccando poi sul bottone *Pay with PayPal* verrà simulato il pagamento di un ordine. Si verrà redirezionati sul sito di test di PayPal dove sarà necessario fare il login con le credenziali di uno degli account di prova impostati nel proprio account PayPal sviluppatore. Se si conferma il pagamento, al ritorno sull'applicazione la lista verrà aggiornata con il nuovo movimento.

7.3 Servizio Facebook

Le funzionalità del servizio sono le seguenti:

- login con Facebook;
- accesso ai post del diario;
- accesso alle foto;
- possibilità di effettuare aggiornamenti di stato;
- pubblicazione foto;
- lettura dei commenti a un aggiornamento di stato o ad una foto.

Per poter utilizzare il servizio è necessario ottenere le credenziali registrando una nuova app sul sito sviluppatori di Facebook. Per creare una nuova applicazione è necessario premere il bottone *Crea Nuova Applicazione* che si trova a destra nella parte alta della pagina. Nella finestra che si aprirà inserire almeno il nome dell'applicazione e confermare premendo su *Continua*.

Creare una nuova app per Facebook

Verrà quindi visualizzata una schermata contenente l'*App ID* e l'*App Secret* relativi all'applicazione appena registrata insieme a tutte le opzioni che è possibile specificare. Le opzioni obbligatorie sono le seguenti:

- se si vuole utilizzare l'applicazione per gestire il login tramite Facebook è necessario indicare nella sezione *Website with Facebook Login* l'URL autorizzato al redirect;
- se si vuole utilizzare l'applicazione per pubblicare aggiornamenti di stato e foto su Facebook è necessario indicare nella sezione *Applicazione su Facebook* l'URL autorizzato al redirect.

L'URL da scrivere, in entrambi i casi, si ottiene da quello dell'applicazione. Nel caso di applicazioni online occorre aggiungere *?CMD=FBAUTH* alla fine, nel caso di applicazioni offline occorre sostituire il nome della pagina con *IDCloud.htm?CMD=FBAUTH*. Ad esempio, se l'URL dell'applicazione è *http://www.progamma.com/Applicazione-/Applicazione.aspx*, l'URL da scrivere sarà *http://www.progamma.com/Applicazione-/IDCloud.htm?CMD=FBAUTH*.

Infine, per salvare le modifiche premere il pulsante *Salva modifiche* alla fine della pagina.

7.3.1 Connessione al server Facebook

Per poter essere in grado di usare il login con Facebook e pubblicare aggiornamenti di stato, link e foto occorre connettersi al server di Facebook e richiedere un *access token*. È necessario perciò creare un oggetto di tipo *Facebook Server* e impostarne le proprietà. Quelle disponibili sono le seguenti:

Client ID	Corrisponde all'App ID assegnato in seguito alla registrazione dell'applicazione sul sito sviluppatori di Facebook
Client Secret	Corrisponde all'App Secret assegnato in seguito alla registrazione dell'applicazione sul sito sviluppatori di Facebook.
Access Token	Valore utilizzato per permettere all'applicazione di accedere alle risorse per conto dell'utente. Se si possiede già un access token valido lo si può indicare usando questo parametro. Non serve nel caso in cui si voglia solamente usare il login con Facebook.
Idform	Id della form che riceverà i messaggi inviati dal componente, se presente.

Per la connessione con il server occorre usare il metodo *Connect* che ha un parametro booleano in cui deve essere specificato se l'applicazione ha un'interfaccia grafica, e un parametro intero in cui devono essere specificati i permessi. Quelli disponibili sono i seguenti:

- Login Only se si vuole utilizzare il login con Facebook
- Read Only se si desidera solo l'accesso in lettura
- Write Only se si desidera solo l'accesso in scrittura
- Read Only + Write Only se si desidera l'accesso in lettura e in scrittura.

Il metodo *Connect* redireziona l'utente sulla pagina Facebook in cui dovrà fare login e poi, la prima volta che si connette, autorizzare Facebook ad accedere al profilo pubblico e alla lista degli amici.

Se la connessione al server non va a buon fine verrà inviato un messaggio "Authorization Error" o "Verification Error" all'Idform e verrà impostato il parametro *Last Error Message*. Se la connessione al server va a buon fine verrà mandato un messaggio "OAUTH" all'Idform e verranno impostate le seguenti proprietà dell'oggetto *Facebook Server*:

User ID	Identificativo univoco dell'utente.
User Name	Nome e cognome dell'utente.
Access Token	Valore utilizzato per permettere all'applicazione di accedere alle risorse per conto dell'utente.

Se, ad esempio, si vuole avere l'accesso in lettura e in scrittura occorre scrivere il seguente codice:

```
public void Form.FacebookConnect()
{
  FacebookServer fs = new()
  fs.ClientID = "985216654233254"
  fs.ClientSecret = "b354462ef2c1f319b0991ca82a412253"
  fs.AccessToken = "CAACGZCju7wm8BAIS86bM1pinIOhFBP0MzBiRcyZAI81UKZAxb5URdB-
          XKDqD1FwNiARxJ2FtJ9rFM1WjL8QkbMD0dHAk9ZBYfXXdLMyZBVWJALRlor4vtz4Kb-
          yq8rmqSIch7cLfGZBkHarFlj6B32R8ZADkTcaVWQGkZD"
  //
  fs.Connect(true, ReadOnly + WriteOnly)
}
```

7.3.2 Login con Facebook

Per gestire il login tramite Facebook occorre connettersi al server impostando le proprietà Client ID e Client Secret, in questo caso non serve impostare la proprietà Access Token. Poi occorre usare il metodo *Connect* con i permessi impostati a LoginOnly, come mostrato nell'immagine seguente:

```
public void FacebookLogin.ButtonLoginWithFacebook()
{
  FacebookServer fs = new()
  fs.ClientID = "985216654233254"
  fs.ClientSecret = "b354462ef2c1f319b0991ca82a412253"
  fs.Idform = this.IDForm()
  //
  fs.Connect(true, LoginOnly)
}
```

Se l'accesso è avvenuto con successo verrà mandato un messaggio "OAUTH" all'Idform e verranno impostate le seguenti proprietà dell'oggetto Facebook Server:

User ID	Identificativo univoco e persistente dell'utente. Può essere usato come chiave (non primaria) per l'identificazione dell'utente.
User Name	Nome e cognome dell'utente

Se, ad esempio, si vuole visualizzare il nome dell'utente dopo il login occorre scrivere il seguente codice:

```
event FacebookLogin.OnSendMessage(
   string Message // Indicates the name of the message
   IDForm Sender  // Identifies the form that sent the message. IDForm type object.
   IDDocument Doc // Optional document to be sent to the sub-form
   string Par1    //
   string Par2    //
   string Par3    //
   string Par4    //
)
{
   if (Message == "OK")
   {
      FacebookServer fs = (FacebookServer)Doc
      //
      LoginPanel.LabelWelcome.text = "Welcome, " + fs.UserName
      LoginPanel.LabelWelcome.setVisible(true)
   }
}
```

7.3.3 Accesso ai post del diario

Per leggere i post che compaiono sul diario occorre caricare la collection *Status Items* dell'oggetto *Facebook Server*. Per ogni oggetto caricato verranno impostate le seguenti proprietà:

ID	Identificativo univoco del post
Message	Il testo del post

Se l'oggetto rappresenta un link verranno impostate anche le seguenti proprietà:

Picture	L'immagine del link
Name	Il titolo del link
Caption	Il sottotitolo del link
Description	Il testo del link

Un esempio di codice è il seguente:

```
public void Form.CommentsFacebook()
{
  // create a Facebook Server object
  FacebookServer fs = new()
  fs.ClientID = "985216654233254"
  fs.ClientSecret = "b354462ef2c1f319b0991ca82a412253"
  fs.AccessToken = "CAACGZCju7wm8BAEG0bTWI506tvdQyWmLRUKD8dkkPPz9uYNcOdfj1x-
          ERH1AtfZClkKczZBWJ5v3YQlUhD5dKN1ZCXnZBjdRswD6D76ZAFje9YzHqqlvXc4i7-
          WZCCZCZCOX3gWgHM4rlxiYVfdoYkeyDoD2AYliyW0s8kZD"
  //
  // connect to the server
  fs.Connect(true, ReadOnly)
  //
  // load status items
  fs.loadCollectionFromDB(fs.StatusItems, ...)
}
```

7.3.4 Accesso alle foto

Per avere accesso alle foto dell'utente occorre caricare la collection *Photos* dell'oggetto *Facebook Server*. Per ogni oggetto caricato verranno impostate le seguenti proprietà:

ID	Identificativo univoco della foto
Message	Descrizione della foto
URL	Link alla foto

Un esempio di codice è il seguente:

```
public void Form.PhotoFacebook()
{
  // create a Facebook Server object
  FacebookServer fs = new()
  fs.ClientID = "985216654233254"
  fs.ClientSecret = "b354462ef2c1f319b0991ca82a412253"
  fs.AccessToken = "CAACGZCju7wm8BAFo59w1z5eZANbVZCZBxNOVZCWJMxYfHzkyreMhUZ-
          AshfsYBzQ6SWgpujPGUpZCmCRUpFnoIgSJeeZBMUaowzGZA39ZAWO0NTDYYAAMfysC-
          0xaEiwQACZAQX6hHmf4qCt1x2ClafIhAZAYL7ZBTZAyNNCHV4ZD"
  //
  // connect to the server
  fs.Connect(true, ReadOnly)
  //
  fs.loadCollectionFromDB(fs.Photos, ...)
}
```

7.3.5 Aggiornamenti di stato

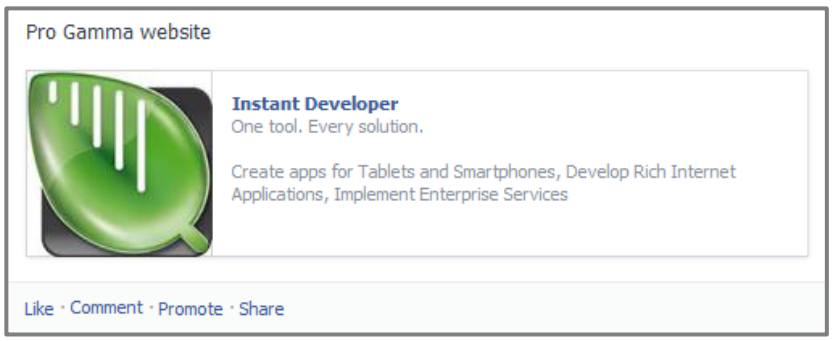
Per pubblicare un nuovo stato occorre creare un oggetto di tipo *Status Item* e impostarne le proprietà. Quelle disponibili sono le seguenti:

Message	Il testo dell'aggiornamento di stato.
Link	L'URL del contenuto da pubblicare.
Picture	L'URL dell'immagine da usare nell'aggiornamento di stato.
Name	Il titolo dell'aggiornamento di stato nel caso di link.
Caption	Sottotitolo dell'aggiornamento di stato nel caso di link.
Description	Descrizione dell'aggiornamento di stato nel caso di link.

Occorre poi aggiungere l'oggetto appena creato alla collection *Status Items* dell'oggetto *Facebook Server*. Se la pubblicazione avviene con successo verrà impostato il parametro ID dell'oggetto Status Item. Se invece si verifica un errore verrà inviato un messaggio "Publish Error" all'Idform e verrà impostato il parametro *Last Error Message*. Se, ad esempio, si vuole pubblicare un link al sito http://www.progamma.com occorre scrivere il seguente codice:

```
public void Form.PostFacebook()
{
  // create a Facebook Server object
  FacebookServer fs = new()
  fs.ClientID = "985216654233254"
  fs.ClientSecret = "b354462ef2c1f319b0991ca82a412253"
  fs.AccessToken = "CAACGZCju7wm8BAB2WeMt7g1NoJoOZAKv5nQbK5TdbrP3OlESsPZBge-
          OndCDgHbsJa4ZCdZB21DtpLC6bJxqFGwxIAThC6D90Btebo6YkamnbAa0rBh7j41xi-
          zc7ghlX90NOjYMESi3INnXxbJl5ig3LwKHm6pqA8ZD"
  //
  // connect to the server
  fs.Connect(true, ReadOnly + WriteOnly)
  //
  // create a Status Item object
  StatusItem si = new()
  si.init()
  si.Message = "Pro Gamma website"
  si.Link = "http://www.progamma.com"
  si.Picture = "https://fbcdn-profile-a.akamaihd.net/hprofile-ak-ash4/c0.0.
          150.150/198988_10150113947794375_4624548_a.jpg"
  si.Name = "Instant Developer"
  si.Caption = "One tool. Every solution."
  si.Description = "Create apps for Tablets and Smartphones, Develop Rich
          Internet Applications, Implement Enterprise Services"
  //
  // add the item to the Status Items collection
  fs.StatusItems.add(si)
  fs.saveToDB([childrenlevel], [revalidate])
}
```

Su Facebook verrà mostrato il seguente link:



7.3.6 Pubblicazione di foto

Per pubblicare una foto occorre usare l'oggetto *Photo Item* e impostarne le proprietà. Quelle disponibili sono le seguenti:

Path	Il path sul server dove si trova la foto
URL	L'URL della foto
Message	Un messaggio sulla foto da pubblicare

Occorre poi aggiungere l'oggetto creato alla collection *Photos* dell'oggetto *Facebook Server*.

Se la pubblicazione avviene con successo verrà impostato il parametro ID dell'oggetto Photo Item. Se invece si verifica un errore verrà inviato un messaggio "Publish Error" all'Idform e verrà impostato il parametro *Last Error Message*.

Se, ad esempio, si vuole pubblicare una foto dal link *http://www.progamma.com/images/progamma_r2_c4_s1.png* occorre scrivere il seguente codice:

```
public void Form.PhotoFacebook()
{
  // create a Facebook Server object
  FacebookServer fs = new()
  fs.ClientID = "985216654233254"
  fs.ClientSecret = "b354462ef2c1f319b0991ca82a412253"
  fs.AccessToken = "CAACGZCju7wm8BAGcnuEFaplMt723TrTUJbTw7aidcTlOmJqgeJIpIu-
          Pfcn0TcMVKDGhzbMGt3V8U87RNhO55O7nZB8JDWGTFZBLloPDZA7WZBEfzAAqtll3k-
          HIvL4ItWo1J8ch7mRkThgV3ywvmVSlx8qogodJUYZD"
  //
  // connect to the server
  fs.Connect(true, WriteOnly)
  //
  // create a Photo Item object
  PhotoItem pi = new()
  pi.init()
  pi.URL = "http://www.progamma.com/images/progamma_r2_c4_s1.png"
  pi.Message = "Instant Developer"
  //
  fs.Photos.add(pi)
  fs.saveToDB(...)
}
```

7.3.7 Lettura commenti

Per leggere i commenti ad un aggiornamento di stato o ad una foto pubblicata occorre creare rispettivamente un oggetto di tipo *Status Item* o un oggetto di tipo *Photo Item*, impostare la proprietà ID e caricare la collection *Comments*. La collection *Comments* verrà riempita con oggetti di tipo *Comment Item* che hanno le seguenti proprietà:

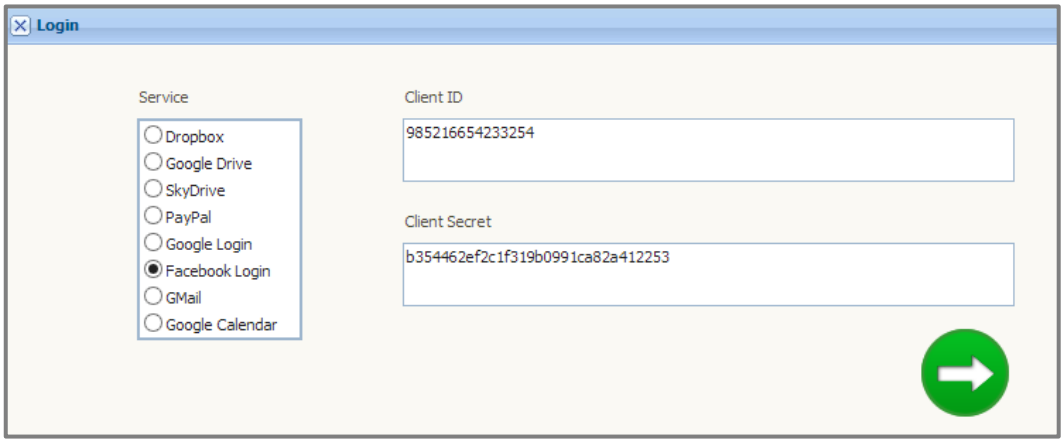
ID	Identificatore univoco del commento.
From	L'utente che ha scritto il commento.
Message	Il testo del commento.
Created Time	La data di pubblicazione del commento.
Like Count	Il numero di like del commento.

Se, ad esempio, si vogliono caricare i commenti relativi a un post con un certo ID occorre scrivere il seguente codice:

```
public void Form.CommentsFacebook()
{
  // create a Facebook Server object
  FacebookServer fs = new()
  fs.ClientID = "985216654233254"
  fs.ClientSecret = "b354462ef2c1f319b0991ca82a412253"
  fs.AccessToken = "CAACGZCju7wm8BAEG0bTWI506tvdQyWmLRUKD8dkkPPz9uYNcOdfj1x-
          ERH1AtfZClkKczZBWJ5v3YQlUhD5dKN1ZCKnZBjdRswD6D76ZAFje9YzHqqlvXc4i7-
          WZCCZCZCOX3gWgHM4rlxiYVfdoYkeyDoD2AYliyW0s8kZD"
  //
  // connect to the server
  fs.Connect(true, ReadOnly)
  //
  // load status items
  fs.loadCollectionFromDB(fs.StatusItems, ...)
  //
  // find the right status item
  for each StatusItem si in fs.StatusItems
  {
    if (si.ID == fs.UserID + "_345869978869378")
    {
      // load comments
      si.loadCollectionFromDB(si.Comments, ...)
      //
      break
    }
  }
}
```

7.3.8 Test del servizio Facebook

Se si desidera testare il servizio Facebook tramite l'applicazione di esempio IDCloud è necessario selezionare Facebook Login nella schermata di login dell'applicazione di esempio e scrivere le credenziali nei rispettivi campi.

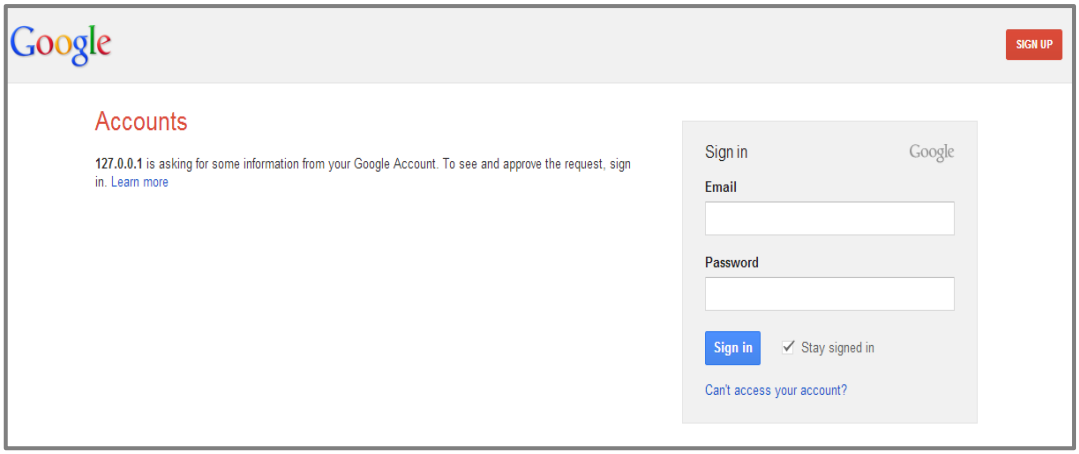


Cliccando sul bottone con la freccia verrà visualizzata la videata che permette di effettuare il login tramite Facebook. Questa videata contiene un solo pulsante *Login with Facebook* che quando premuto reindirizza il browser alla pagina di login di Facebook dove sarà necessario inserire prima email e password e poi autorizzare l'accesso ai dati. Dopo aver cliccato su "Accetto" il browser ritornerà all'applicazione in una videata che riepiloga i dati disponibili dell'account.

7.4 Servizio Google Profile

Il servizio Google Profile permette di integrare il login di Google nelle applicazioni realizzate con Instant Developer. Per poterlo utilizzare occorre creare un oggetto di tipo *Google Profile* e poi impostare la proprietà Idform per indicare la videata che riceverà i messaggi.

Usando il metodo *Connect* l'utente viene redirezionato sulla pagina di Google in cui dovrà fare login e, la prima volta che si connette, concedere a Google l'autorizzazione all'utilizzo dei dati dell'account (nome, sesso, data di nascita, nazione, lingua) e dell'indirizzo email.



Un esempio di codice è il seguente:

```
public void GoogleLogin.ButtonLoginWithGoogle()
{
   GoogleProfile gp = new()
   gp.Idform = this.IDForm()
   gp.Connect(true)
}
```

Se l'utente concede l'autorizzazione verrà inviato un messaggio "OAUTH" all'Idform e verranno impostate le seguenti proprietà dell'oggetto Google Profile:

ID	Identificativo univoco e persistente dell'utente. Può essere usato come chiave per l'identificazione dell'utente.
Name	Nome dell'utente
Surname	Cognome dell'utente
Email	Indirizzo email dell'utente
Country	Paese dell'utente
Language	Lingua dell'utente

Se, ad esempio, si vogliono visualizzare i dati dell'utente dopo il login occorre scrivere il seguente codice:

```
event GoogleLogin.OnSendMessage(
   string Message // Indicates the name of the message
   IDForm Sender  // Identifies the form that sent the message. IDForm type object.
   IDDocument Doc // Optional document to be sent to the sub-form
   string Par1    //
   string Par2    //
   string Par3    //
   string Par4    //
)
{
   if (Message == "OK")
   {
      GoogleProfile gp = (GoogleProfile)Doc
      //
      LoginPanel.LabelWelcome.text = "Welcome, " + gp.Name + " " + gp.Surname +
               "<br/><br/>Email: " + gp.Email + "<br/>Language: " + gp.Language +
               "<br/>Country: " + gp.Country
      LoginPanel.LabelWelcome.setVisible(true)
   }
}
```

Per provare il servizio Google Profile nell'applicazione di esempio IDCloud è necessario selezionare Google Login nella schermata iniziale. A questo punto verrà visualizzata una videata che contiene un solo pulsante "Login with Google".

Premendo questo bottone si verrà reindirizzati alla pagina di login di Google dove sarà necessario inserire prima email e password e poi autorizzare l'accesso ai dati dell'account e alla mail. Dopo aver cliccato su "Accetto" si verrà reindirizzati ad una videata dell'applicazione che riepiloga i dati dell'account.

7.5 Servizio Google Calendar

Il servizio Google Calendar permette di integrare il calendario di Google all'interno delle applicazioni realizzate con Instant Developer. Le funzionalità del servizio sono le seguenti:

- lista dei calendari;
- visualizzazione di eventi;
- creazione, modifica, cancellazione di eventi.

Per poter utilizzare il servizio è necessario ottenere le credenziali registrando una nuova applicazione al sito https://code.google.com/apis/console. Se si sta registrando un'applicazione per la prima volta, si presenterà la seguente videata:

In caso contrario si aprirà direttamente la console per la gestione delle applicazioni. Per registrare una nuova applicazione occorre compiere le seguenti azioni:

- Se è la prima volta che si accede alla console sviluppatori, premere il pulsante *Create project…*
- Se non è la prima volta, cliccare sul menu a tendina *API Project* e scegliere *Create*. Nella finestra che si aprirà, inserire il nome del progetto e premere il pulsante *Create project*.

- Scegliere *Services* dal menu sulla sinistra e abilitare le *Calendar API* cliccando sul pulsante OFF e accettando poi i termini del servizio.
- Scegliere *API Access* dal menu sulla sinistra e premere il pulsante *Create an OAuth 2.0 client ID…*
- Nella finestra che si aprirà, scrivere almeno il *Product name* cliccare su *Next*.
- Nella finestra successiva scegliere *Web Application* come tipo di applicazione.
- Cliccare su *More option* e scrivere nello spazio relativo agli *Authorized Redirect URIs* l'URL autorizzato al redirect. Nel caso di applicazioni online, esso si ottiene aggiungendo alla fine dell'URL dell'applicazione *?CMD=AUTHCAL* mentre, nel caso di applicazioni offline, si ottiene dall'URL dell'applicazione sostituendo il nome della pagina con *IDCloud.htm?CMD=AUTHCAL*. Ad esempio, se l'URL dell'applicazione è *http://www.progamma.com/Applicazione/Applicazione.aspx*, l'URL da scrivere sarà *http://www.progamma.com/Applicazione/-IDCloud.htm?CMD=AUTHCAL*. Può essere specificato più di un URL.
- Premere il bottone *Create client ID* per ottenere le credenziali d'accesso.
- Le credenziali d'accesso sono indicate nella pagina *API Access* nell'area *Client ID for web applications*.

7.5.1 Connessione al server Google

Per poter essere in grado di usare i servizi di Google occorre utilizzare le credenziali ottenute al momento della registrazione dell'applicazione per richiedere un access token e un refresh token. È necessario perciò creare un oggetto di tipo *Calendar Server* e impostarne le proprietà. Quelle disponibili sono le seguenti:

Client ID	Valore assegnato in seguito alla registrazione dell'applicazione sulla console di Google.
Client Secret	Valore assegnato in seguito alla registrazione dell'applicazione sulla console di Google.
Access Token	Valore utilizzato per permettere all'applicazione di accedere alle risorse per conto dell'utente. Se si possiede già un access token valido lo si può indicare usando questo parametro.
Refresh Token	Valore utilizzato per ottenere un nuovo Access Token quando quest'ultimo scade. Se si possiede già un refresh token valido lo si può indicare usando questo parametro.
Idform	Id della form che riceverà i messaggi inviati dal componente, se presente.

Per la connessione con il server occorre usare il metodo *Connect* che ha un parametro booleano con il quale deve essere specificato se l'applicazione ha un'interfaccia grafica e un altro parametro booleano che deve essere impostato a true se si desidera l'accesso in sola lettura.

Il metodo *Connect* redireziona l'utente sulla pagina Google in cui dovrà fare login e poi, la prima volta che si connette, autorizzare Google ad accedere ai dati dell'utente. Se la connessione al server non va a buon fine verrà inviato un messaggio "Authorization Error" o "Verification Error" all'Idform e verrà impostato il parametro *Last Error Message*. Se la connessione va a buon fine verrà mandato un messaggio "OAUTH" all'Idform e verranno impostate le seguenti proprietà dell'oggetto *Calendar Server*.

User ID	Identificativo univoco dell'utente.
User Name	Nome e cognome dell'utente.
Access Token	Valore utilizzato per permettere all'applicazione di accedere alle risorse per conto dell'utente.
Refresh Token	Valore utilizzato per recuperare una nuova Access Token dopo che quest'ultima scade.

L'*Access Token* ha un tempo di vita limitato indicato dalla proprietà *Expires In* dell'oggetto *Calendar Server*. Tuttavia l'oggetto *Calendar Server* è in grado di recuperare in modo automatico un nuovo *Access Token* quando quello vecchio scade preoccupandosi quindi di mantenere attiva la connessione.

Il *Refresh Token* viene ottenuto soltanto la prima volta che l'utente si connette al servizio, è importante quindi salvarlo al momento della prima connessione. Per ottenerne uno nuovo occorre chiedere all'utente di revocare le autorizzazioni concesse all'applicazione, in questo modo quando l'utente si riconnetterà Google invierà un altro *Refresh Token.* Le operazioni che l'utente deve eseguire per revocare le autorizzazioni sono le seguenti:

- Accedere al proprio account.
- Scegliere la tab *Sicurezza*.
- Cliccare su *Visualizza tutto* nel riquadro *Autorizzazioni account.*
- Selezionare l'applicazione dalla lista.
- Cliccare sul bottone *Revoca accesso* che si trova sulla destra.

Se, ad esempio, si vuole ottenere l'accesso in lettura e scrittura occorre scrivere il seguente codice:

```
public void Form.CalendarConnection()
{
  // create a Calendar Server object
  CalendarServer cs = new()
  cs.ClientID = "365482156985.apps.googleusercontent.com"
  cs.ClientSecret = "JEPDvERicvUODnuOfUncdOPG"
  cs.AccessToken = "ya29.AHES6ZRT17UilMLtTA1VY4E8XDOgFyoMHvxl2Rlx1iVhcVrCfgmaYp0M"
  cs.RefreshToken = "1/Pyn2DF_sGmlZ_zYnmQQaQxK0di6ZpUPEn6yyLu3VEaA"
  //
  // connection to the server
  cs.Connect(true, false)
}
```

7.5.2 Ottenere la lista dei calendari

Per ottenere la lista dei calendari dell'utente occorre caricare la collection *Calendars* dell'oggetto *Calendar Server*. Gli oggetti caricati sono di tipo *Calendar Item* e, per ognuno di essi, verranno impostate le seguenti proprietà:

ID	Identificativo univoco del calendario.
Name	Nome del calendario.
Time Zone	Time zone del calendario.
Colorid	Id del colore associato al calendario.

Un esempio di codice è il seguente:

```
public void Form.CalendarConnection()
{
  // create a Calendar Server object
  CalendarServer cs = new()
  cs.ClientID = "365482156985.apps.googleusercontent.com"
  cs.ClientSecret = "JEPDvERicvUODnuOfUncdOPG"
  cs.AccessToken = "ya29.AHES6ZRT17UilMLtTA1VY4E8XDOgFyoMHvxl2Rlx1iVhcVrCfgmaYp0M"
  cs.RefreshToken = "1/Pyn2DF_sGmlZ_zYnmQQaQxK0di6ZpUPEn6yyLu3VEaA"
  //
  // connection to the server
  cs.Connect(true, false)
  //
  // load calendars
  cs.loadCollectionFromDB(cs.Calendars, [childrenlevel])
}
```

7.5.3 Ottenere la lista degli eventi

Per ottenere la lista degli eventi di un calendario occorre caricare la collection *Events* dell'oggetto *Calendar Item*. Gli oggetti caricati sono di tipo *Event Item* e, per ognuno di essi, verranno impostate le seguenti proprietà:

ID	Identificativo univoco del calendario.
Name	Titolo dell'evento.
Description	Descrizione dell'evento.
Location	Posizione geografica dell'evento.
Start	Data e ora di inizio dell'evento.
End	Data e ora di fine dell'evento.
Sequence	Numero di revisione
Recurring Event ID	Se l'evento è un'istanza di un evento ricorrente, è l'ID dell'evento ricorrente generatore.
Color ID	ID del colore associato all'evento
Reminder	Tipo di promemoria. I valori possibili sono: email, sms, popup.
Reminder Time	Quanti minuti prima dell'evento verrà inviato il promemoria.
Updated	Data e ora dell'ultima modifica dell'evento.
Private Properties	Proprietà private dell'evento configurabili dall'utente.
Shared Properties	Proprietà configurabili dall'utente condivise con le copie dell'evento presenti in altri calendari.

In questo modo, verranno caricati tutti gli eventi del calendario ordinati per data e ora di inizio. Se invece si vogliono ottenere solo gli eventi in un determinato intervallo di tempo, occorre impostare le proprietà *Start* e *End* dell'oggetto *Calendar Item*.

Se ci sono eventi ricorrenti, si otterranno tutte le istanza dell'evento e non soltanto l'evento generatore. Se, ad esempio, si vogliono ottenere tutti gli eventi del calendario *My Calendar* occorre scrivere il seguente codice:

```
public void Form.CalendarConnection()
{
  // create a Calendar Server object
  CalendarServer cs = new()
  cs.ClientID = "365482156985.apps.googleusercontent.com"
  cs.ClientSecret = "JEPDvERicvUODnuOfUncdOPG"
  cs.AccessToken = "ya29.AHES6ZRT17UilMLtTA1VY4E8XDOgFyoMHvxl2Rlx1iVhcVrCfgmaYp0M"
  cs.RefreshToken = "1/Pyn2DF_sGmlZ_zYnmQQaQxK0di6ZpUPEn6yyLu3VEaA"
  //
  // connection to the server
  cs.Connect(true, false)
  //
  // load calendars
  cs.loadCollectionFromDB(cs.Calendars, ...)
  //
  // search "My Calendar" calendar
  for each CalendarItem ci in cs.Calendars
  {
    if (ci.Name == "My Calendar")
    {
      // load events
      ci.loadCollectionFromDB(ci.Events, [childrenlevel])
    }
    //
    break
  }
}
```

7.5.4 Creazione di un nuovo evento

Per aggiungere un nuovo evento occorre creare un oggetto di tipo *Event Item* e impostarne le proprietà. Quelle disponibili sono le seguenti:

Name	Titolo dell'evento.
Description	Descrizione dell'evento.
Location	Posizione geografica dell'evento.
Start	Data e ora di inizio dell'evento.
End	Data e ora di fine dell'evento.
Color ID	ID del colore associato all'evento.
Reminder	Tipo di promemoria. I valori possibili sono: email, sms, popup.
Reminder Time	Quanti minuti prima dell'evento verrà inviato il promemoria.
Frequence	Con quale frequenza si deve ripetere l'evento. I valori possibili sono: Daily, Weekly, Monthly, Yearly
Interval	Quanto spesso si deve ripetere l'evento.
Until Date	Fino a che data si deve ripetere l'evento.
Until Occurences	Quante volte si deve ripetere l'evento.

Repeat On	Quali giorni della settimana si deve ripetere l'evento.

Occorre poi aggiungere l'oggetto *Event Item* creato alla collection *Events* dell'oggetto *Calendar Item*. Dopo aver creato in questo modo tutti gli eventi che si desidera aggiungere al calendario occorre salvare la collection *Events* dell'oggetto *Calendar Item*.

Se, ad esempio, nel calendario *My Calendar* si vuole creare un evento che deve ripetersi ogni 3 giorni, dalle 10.00 alle 11.00, dal 21 Maggio fino al 21 Giugno con un promemoria via email 10 minuti prima, occorre scrivere il seguente codice:

```
public void Form.CalendarConnection()
{
  // create a Calendar Server object
  CalendarServer cs = new()
  cs.ClientID = "365482156985.apps.googleusercontent.com"
  cs.ClientSecret = "JEPDvERicvUODnuOfUncdOPG"
  cs.AccessToken = "ya29.AHES6ZRT17UilMLtTA1VY4E8XDOgFyoMHvxl2Rlx1iVhcVrCfgmaYp0M"
  cs.RefreshToken = "1/Pyn2DF_sGmlZ_zYnmQQaQxK0di6ZpUPEn6yyLu3VEaA"
  //
  // connection to the server
  cs.Connect(true, false)
  //
  // load calendars
  cs.loadCollectionFromDB(cs.Calendars, ...)
  //
  // search "My Calendar" calendar
  for each CalendarItem ci in cs.Calendars
  {
    if (ci.Name == "My Calendar")
    {
      // create an Event Item object
      EventItem ei = new()
      ei.init()
      ei.Name = "Meeting with Mr. Smith"
      ei.Start = toDateTime("21/05/2013 10:00")
      ei.End = toDateTime("21/05/2013 11:00")
      //
      // set the reminder properties
      ei.Reminder = email
      ei.ReminderTime = 10
      //
      // set the recurrence properties
      ei.Frequence = Daily
      ei.Interval = 3
      ei.UntilDate = toDateTime("21/06/2013")
      ..
    // add the event to the collection of the calendar "My Calendar"
      ci.Events.add(ei)
      ci.Events.saveToDB([childrenlevel], [revalidate])
    }
    //
    break
  }
}
```

7.5.5 Modifica di un evento

Per modificare un evento precedentemente inserito in un calendario occorre recuperare l'evento e agire sulle proprietà che si vogliono cambiare. Dopo aver modificato tutti gli eventi desiderati, occorre salvare la collection *Events* che li contiene.

Se, ad esempio, si vuole spostare l'evento il cui ID è h104m268294uekunrqbp1n9pvs_20130218T110000Z al 25 maggio 2013 dalle 10:00 alle 11:00 occorre scrivere il seguente codice:

```
public void Form.ModifyEvent()
{
  // create a Calendar Server object
  CalendarServer cs = new()
  cs.ClientID = "365482156985.apps.googleusercontent.com"
  cs.ClientSecret = "JEPDvERicvUODnuOfUncdOPG"
  cs.AccessToken = "ya29.AHES6ZRhR0SfBT4aTb6cDBHoCZuQa7kNYhEsQZ2098YdGUvVWQ6cTlJQ"
  cs.RefreshToken = "1/Pyn2DF_sGmlZ_zYnmQQaQxK0di6ZpUPEn6yyLu3VEaA"
  //
  // connection to the server
  cs.Connect(true, false)
  //
  // load calendars
  cs.loadCollectionFromDB(cs.Calendars, ...)
  //
  // search "My Calendar" calendar
  for each CalendarItem ci in cs.Calendars
  {
    if (ci.Name == "My Calendar")
    {
      ci.loadCollectionFromDB(ci.Events, ...)
      //
      // search the event
      for each EventItem ei in ci.Events
      {
        if (ei.ID == "h104m268294uekunrqbp1n9pvs_20130218T110000Z")
        {
          // modify the date
          ei.Start = toDateTime("25/05/2013 10:00")
          ei.End = toDateTime("25/05/2013 11:00")
          //
          //
          // save the event
          ci.Events.saveToDB(...)
          //
          break
        }
      }
      //
      break
    }
  }
}
```

7.5.6 Cancellazione un evento

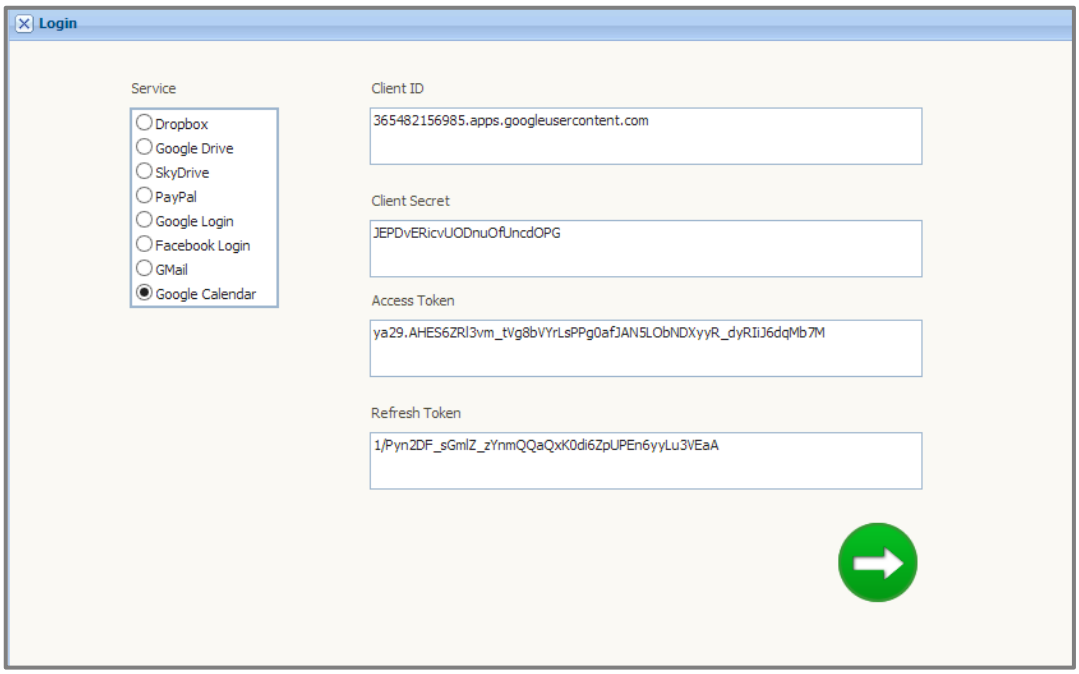
Per eliminare un evento è necessario marcarlo per la cancellazione impostando a *true* la proprietà *Deleted*. Dopo aver eliminato tutti gli eventi che si desiderava cancellare occorre salvare la collection *Events* che li contiene.

Se, ad esempio, si vuole cancellare l'evento il cui ID è h104m268294uekunrqbp1n9pvs_20130218T110000Z è necessario scrivere il seguente codice:

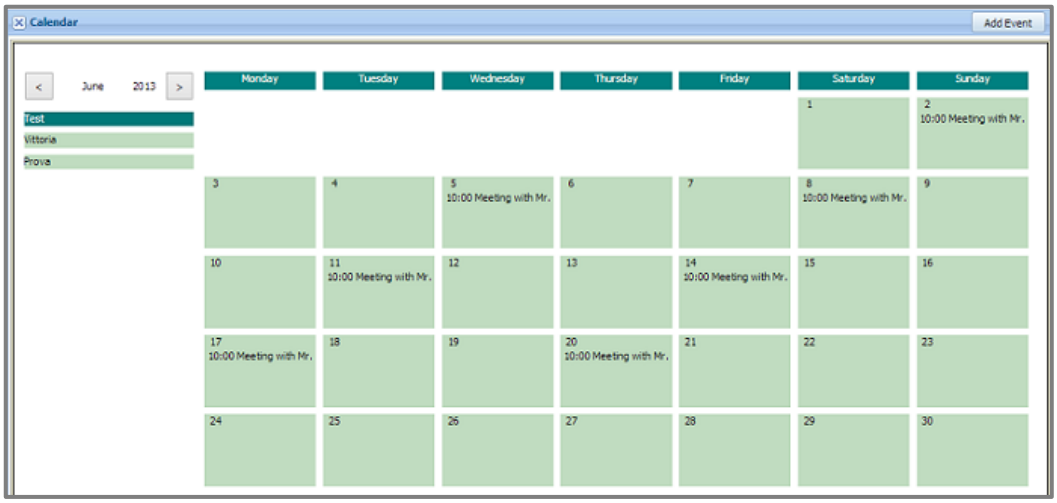
```
public void Form.DeleteEvent1()
{
  // create a Calendar Server object
  CalendarServer cs = new()
  cs.ClientID = "365482156985.apps.googleusercontent.com"
  cs.ClientSecret = "JEPDvERicvUODnuOfUncdOPG"
  cs.AccessToken = "ya29.AHES6ZRhR0SfBT4aTb6cDBHoCZuQa7kNYhEsQZ2098YdGUvVWQ6cTlJQ"
  cs.RefreshToken = "1/Pyn2DF_sGmlZ_zYnmQQaQxK0di6ZpUPEn6yyLu3VEaA"
  //
  // connection to the server
  cs.Connect(true, false)
  //
  // load calendars
  cs.loadCollectionFromDB(cs.Calendars, ...)
  //
  // search "My Calendar" calendar
  for each CalendarItem ci in cs.Calendars
  {
    if (ci.Name == "My Calendar")
    {
      ci.loadCollectionFromDB(ci.Events, ...)
      //
      // search the event
      for each EventItem ei in ci.Events
      {
        if (ei.ID == "h104m268294uekunrqbp1n9pvs_20130218T110000Z")
        {
          // delete the event
          ei.deleted = true
          ci.Events.saveToDB(...)
          //
          break
        }
      }
      //
      break
    }
  }
}
```

7.5.7 Test del servizio Google Calendar

Per provare il servizio *Google Calendar* tramite l'applicazione di esempio IDCloud è necessario selezionare Google Calendar nella schermata iniziale dell'applicazione e scrivere le credenziali nei rispettivi campi.

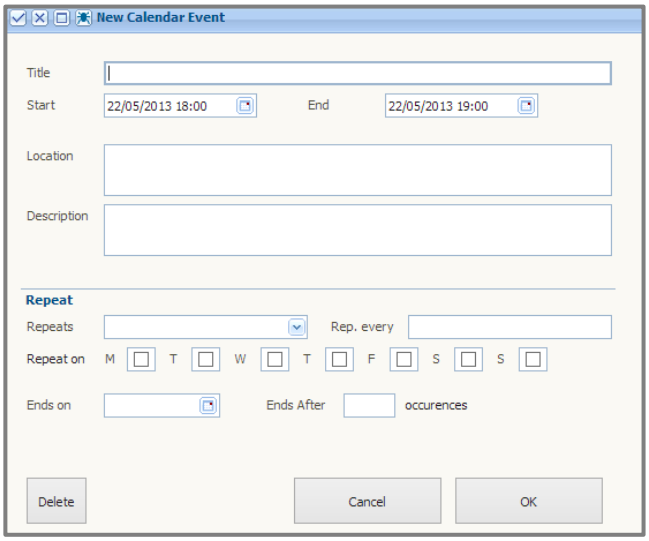


Cliccando sul bottone con la freccia verrà visualizzata la videata che mostra il calendario relativo al mese corrente.



Sulla sinistra è presente un elenco dei calendari dell'utente. Per vedere gli eventi relativi a un calendario occorre cliccare su di esso. Sopra la lista dei calendari verrà visualizzato il nome del mese a cui la videata si riferisce e l'anno. Cliccando sulle frecce accanto al nome del mese è possibile navigare nel calendario.

Per aggiungere un nuovo evento occorre premere il bottone *Add Event* che si trova nell'angolo in alto a destra. Dopo aver riempito opportunamente i campi, premere OK per aggiungere l'evento al calendario. Per modificare un evento occorre fare doppio click su di esso. Nella videata che si aprirà occorre modificare tutti i campi che si desidera cambiare e premere OK per salvare le modifiche. Per eliminare un evento occorre fare doppio click su di esso e poi premere il bottone *Delete* in basso a sinistra.



7.6 Servizio GMail

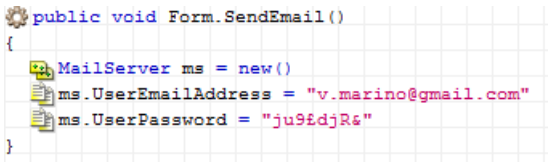
Il servizio *GMail* permette di leggere e inviare email con *GMail* o con un altro provider di posta all'interno delle applicazioni realizzate con Instant Developer. Le funzionalità del servizio sono le seguenti:

- Lettura delle email.
- Download allegati delle email.
- Invio di email.

Per connettersi al server occorre creare un oggetto di tipo *Mail Server* e impostare le seguenti proprietà:

User Email Address	Indirizzo email dell'utente.
User Password	Password dell'utente.
IncomingHost	(opzionale) Host per la ricezione delle email. Se non viene impostato viene utilizzato quello di Gmail.
IncomingPort	(opzionale) Porta per la ricezione delle email. Se non viene impostata viene utilizzata la porta 993.
IncomingTLS	(opzionale) Se impostato a false non viene usato il protocollo TLS per la ricezione delle email. Il valore di default è true.
OutgoingHost	(opzionale) Host per l'invio delle email. Se non viene impostato viene utilizzato quello di Gmail.
OutgoingPort	(opzionale) Porta per l'invio delle email. Se non viene impostata viene utilizzata la porta 587.
OutgoingTLS	(opzionale) Se impostato a false non viene usato il protocollo TLS per l'invio delle email. Il valore di default è true.
Download Full Email	(opzionale) Se impostata a true indica al metodo *ReadEmails* che deve recuperare le email per intero, altrimenti vengono recuperati solo gli headers. Il valore di default è false.

Un esempio di codice è il seguente:



7.6.1 Lettura delle email

Le email sono contenute in una o più mailbox, perciò per poterle leggere è necessario prima di tutto ottenere la lista delle mailbox. Ogni mailbox è rappresentata da un oggetto di tipo *MailboxItem* che possiede le seguenti proprietà:

Name	Nome della mailbox.
Last Read Email ID	Identificativo univoco dell'ultima email presente nella mailbox. Questa proprietà viene valorizzata quando si popola la collection *Mails* della mailbox.

Ogni oggetto di tipo *MailboxItem* può contenere altri oggetti dello stesso tipo che rappresentano le sue sottomailbox e oggetti di tipo *MailItem* che rappresentano le email. Per ottenere le mailbox occorre usare la funzione *GetMailbox* che riempie la collection *Mailboxes* dell'oggetto *MailServer* e di ogni oggetto *MailboxItem* caricato.

Per ottenere le email contenute in una mailbox occorre popolare la collection *Mails* dell'oggetto *Mailbox Item*. Verranno recuperate le email (o solo gli header se la proprietà *Download Full Email* è impostata a false) successive a quella con ID uguale al valore di *Last Read Email ID* e verrà aggiornato il valore di questa proprietà.

Se si vogliono ottenere solo un certo numero di email, si può specificare il parametro *MaxRows* della collection *Mails*. Verranno in questo modo recuperate le ultime *MaxRows* email andando indietro non oltre quella con ID uguale al valore *Last Read Email ID*. Se si vogliono ottenere anche email già recuperate in precedenza, è possibile modificare *Last Read Email ID*. L'ID della prima email è sempre uguale a 1.

La collection *Mails* verrà popolata da oggetti di tipo *MailItem* per i quali verranno impostate le seguenti proprietà:

ID	Identificativo univoco dell'email.
From	Mittente dell'email.
To	Destinatari dell'email.
Cc	Destinatari in copia conoscenza.
Bcc	Destinatari in copia conoscenza nascosta.
Reply To	Indirizzo email al quale devono essere inviate le risposte, se diverso da quello del mittente
Subject	Oggetto dell'email.
Html Body	Corpo dell'email in formato HTML. Non è presente nel caso si siano scaricati solo gli headers delle email.
Text Body	Corpo dell'email in formato testo. Non è presente nel caso si siano scaricati solo gli headers delle email.
Date	Data di ricevimento dell'email.

Se sono stati scaricati solo gli headers delle email, per ottenere l'email completa occorre utilizzare la funzione *CompleteEmail* dell'oggetto *MailItem*. Se l'operazione fallisce, la funzione restituisce una stringa che descrive l'errore. In caso contrario, vengono impostate le proprietà dell'oggetto *Mail Item* e viene popolata la collection *Attachments* con gli allegati dell'email, se presenti. La funzione *CompleteEmail* può essere usata anche per recuperare un'email a partire dall'id. In questo caso occorre creare un nuovo oggetto di tipo *MailItem,* impostare la proprietà *ID* e aggiungerlo alla collection Mails dell'oggetto *MailboxItem* nel quale cercare l'email. Chiamando la funzione *CompleteEmail,* verranno valorizzate le proprietà dell'oggetto *MailItem* appena creato.

Se, ad esempio, si vogliono recuperare le prime 5 email della mailbox “INBOX” e ottenere per intero l’ultima email ricevuta occorre scrivere il seguente codice.

```
public void Form.GetEmail()
{
   MailServer ms = new()
   ms.UserEmailAddress = "v.marino@gmail.com"
   ms.UserPassword = "ju9£djR&"
   //
   // load mailboxes
   ms.GetMailboxes("")
   //
   // search the INBOX mailbox
   for each MailboxItem mailbox in ms.Mailboxes
   {
      if (mailbox.Name == "INBOX")
      {
         // set the number on emails to get
         mailbox.Mails.maxRows = 5
         //
         // load emails
         mailbox.loadCollectionFromDB(mailbox.Mails, ...)
         //
         // get the last email
         for each MailItem mail in mailbox.Mails
         {
            if (mail.ID == mailbox.LastReadEmailID)
            {
               mail.GetSingleEmail()
               //
               break
            }
         }
         break
      }
   }
   //
   //

}
```

7.6.2 Download di un allegato

Per ottenere gli allegati di un’email occorre caricare la collection *Attachments* dell’oggetto *MailItem*. Per scaricare un allegato occorre usare la funzione *Download* che possiede un parametro stringa che rappresenta il percorso dove salvare il file. Se l’operazione non va a buon fine la funzione restituirà il testo dell’errore e imposterà il parametro Last Error Message.

Se, ad esempio, si vuole scaricare l’allegato dell’ultima email arrivata nella cartella *temp* dell’applicazione, occorre scrivere il seguente codice:

```
public void Form.GetAttachments()
{
   MailServer ms = new()
   ms.UserEmailAddress = "v.marino@gmail.com"
   ms.UserPassword = "ju9£djR&"
   //
   // load mailboxes
   ms.GetMailboxes("")
   //
   // search the INBOX mailbox
   for each MailboxItem mailbox in ms.Mailboxes
   {
      if (mailbox.Name == "INBOX")
      {
         // set the number on emails to get
         mailbox.Mails.maxRows = 5
         //
         // load emails
         mailbox.loadCollectionFromDB(mailbox.Mails, ...)
         //
         // get the last email
         for each MailItem mail in mailbox.Mails
         {
            if (mail.ID == mailbox.LastReadEmailID)
            {
               mail.loadCollectionFromDB(mail.Attachments, ...)
               //
               for each AttachmentItem ai in mail.Attachments
               {
                  ai.Download(NuovaApplicazioneWeb.path() + "/temp")
               }
               //
               break
            }
         }
         break
      }
   }
}
```

7.6.3 Invio di un'email

Per inviare un'email occorre creare un oggetto di tipo *MailItem* e impostarne le proprietà. Quelle disponibili sono le seguenti:

From	Mittente dell'email.
To	Destinatari dell'email.
Cc	Destinatari in copia conoscenza.
Bcc	Destinatari in copia conoscenza nascosta.
Reply To	Indirizzo email al quale devono essere inviate le risposte, se diverso da quello del mittente

Subject	Oggetto dell'email.
Html Body	Contenuto della mail in formato HTML.
Text Body	Contenuto della mail in formato testo.
Save Only	Deve essere impostato a *true* se si vuole salvare l'email in una mailbox ma non inviarla. Il valore di default è *false*.

Se si vogliono mandare anche degli allegati, occorre aggiungerli alla collection *Attachments* dell'oggetto *MailItem*. Occorre poi aggiungere l'oggetto così creato alla collection *Mails* di un oggetto *MailboxItem*, l'email verrà salvata in questa mailbox.
Un esempio di codice è il seguente:

```
public void Form.SendEmail()
{
  MailServer ms = new()
  ms.UserEmailAddress = "v.marino@gmail.com"
  ms.UserPassword = "ju9$djR&"
  //
  // load mailboxes
  ms.GetMailboxes("")
  //
  // create a Mail Item object
  MailItem email = new()
  email.init()
  email.To.addString("v.marino@progamma.com")
  email.To.addString("info@progamma.com")
  email.Cc.addString("support@progamma.com")
  email.Subject = "Info"
  email.HtmlBody = "Hello world!"
  //
  // add the email to a mailbox and save it
  MailboxItem mailbox = (MailboxItem)ms.Mailboxes.getDirect(1)
  mailbox.Mails.add(email)
  mailbox.Mails.saveToDB([childrenlevel], [revalidate])
}
```

7.6.3 Spostamento di un'email da una mailbox a un'altra

Per spostare un'email da una mailbox a un'altra occorre chiamare la funzione *MoveEmail* dell'oggetto *MailServer* passando come parametri il nome della mailbox di destinazione e l'oggetto *MailItem* che si vuole spostare. Affinchè lo spostamento vada a buon fine, l'oggetto *MailItem* deve appartenere alla collection *Mails* del *MailboxItem* sorgente. Se l'operazione non va a buon fine la funzione restituirà il testo dell'errore e imposterà il parametro Last Error Message, altrimenti verrà restituito il nuovo id assegnato all'email dopo lo spostamento.
Se, per esempio, si vuole spostare un'email dalla mailbox *INBOX* alla mailbox *Deleted Items*, occorre scrivere il seguente codice:

```
public void NewForm.ButtonMoveEmail()
{
  this.MailServer.loadCollectionFromDB(MailServer.Mailboxes, ...)
  //
  MailItem myEmail = null
  for each MailboxItem mi in MailServer.Mailboxes
  {
    //
    if (mi.Name == "INBOX")
    {
      mi.loadCollectionFromDB(mi.Mails, ...)
      //
      for each MailItem mail in mi.Mails
      {
        if (mail.ID == "598")
        {
          myEmail = mail
          break
        }
      }
    }
  }
  //
  string res = this.MailServer.MoveEmail("Deleted Items", myEmail)
}
```

7.7 Servizio Dropbox

Il servizio Dropbox permette di gestire i file su Dropbox all'interno delle applicazioni realizzate con Instant Developer. Le funzionalità del servizio sono le seguenti:

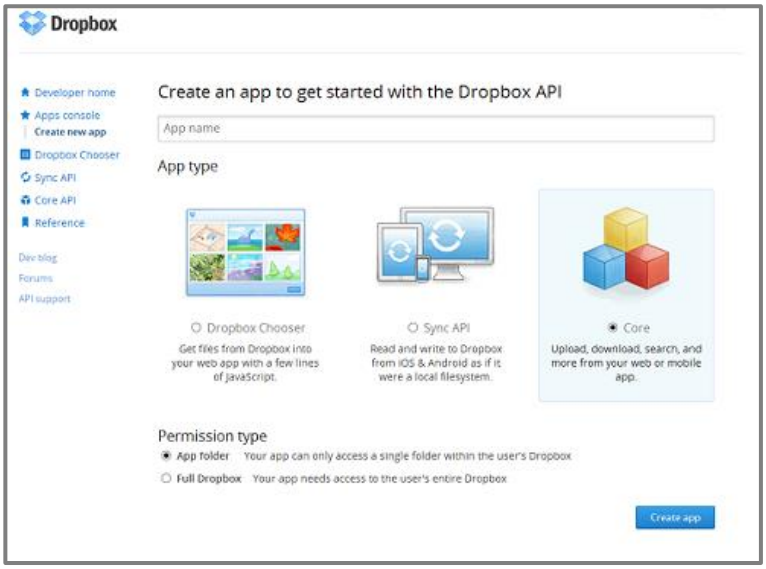
- creazione cartelle;
- lista dei file e directory presenti sul server;
- upload e download di file;
- eliminazione di file e cartelle;
- copia e spostamento di file da una cartella a un'altra.

Per poter utilizzare il servizio è necessario essere in possesso delle credenziali che si ottengono registrando una nuova applicazione sul sito sviluppatori di Dropbox.

7.7.1 Richiesta credenziali

Per registrare una nuova applicazione occorre cliccare su *Apps console* > *Create new app* nel menu laterale di sinistra, nella nuova pagina compiere le seguenti operazioni:

- scrivere il nome della nuova applicazione nell'apposito spazio;
- scegliere *Core* come tipo di applicazione;
- scegliere il tipo di permessi da assegnare all'applicazione;
- premere il bottone *Create app*.



Se l'applicazione viene creata con successo, verrà mostrata una pagina di riepilogo dei dati relativi ad essa, tra cui anche l'App key e l'App secret che dovranno essere impostati rispettivamente nelle proprietà *Client ID* e *Client Secret* dell'oggetto *Drive Server*.

7.7.2 Connessione al server

Per poter effettuare qualsiasi operazione su file e cartelle occorre prima essere connessi al server Dropbox. Per connettersi è necessario creare un oggetto di tipo *Drive Server* e impostare le seguenti proprietà:

Client ID	Corrisponde all'App key assegnato in seguito alla registrazione dell'applicazione sul sito sviluppatori di Dropbox.
Client Secret	Corrisponde all'App secret assegnato in seguito alla registrazione dell'applicazione sul sito sviluppatori di Dropbox.
Secret Token	Valore utilizzato per stabilire la corrispondenza tra Access Token e Client ID
Server Type	Indica a quale server ci si vuole connettere. I valori possibili sono: Dropbox, GoogleDrive e SkyDrive
Idform	Id della form che riceverà i messaggi inviati dal componente, se presente.
Read Only	Deve essere impostato a *true* se si vuole accedere al server in sola

	lettura

Occorre quindi usare il metodo *Connect* che ha un parametro booleano con il quale deve essere specificato se l'applicazione ha un'interfaccia grafica e un altro parametro booleano che deve essere impostato a true se si desidera l'accesso in sola lettura.

Il metodo *Connect* redireziona l'utente sulla pagina Dropbox in cui dovrà fare login e poi, la prima volta che si connette, autorizzare Dropbox ad accedere ai suoi dati. Se la connessione al server non va a buon fine verrà inviato un messaggio "Authorization Error" o "Verification Error" all'Idform e verrà impostato il parametro *Last Error Message.* Se la connessione al server va a buon fine verrà inviato un messaggio "OAUTH" all'Idform, verrà impostata la *Root Directory* e la proprietà *User Name* dell'oggetto *Drive Server.*

Se, ad esempio, si vuole accedere al server in lettura e scrittura occorre scrivere il seguente codice:

```
public void Form.DropboxConnect()
{
  // create a Drive Server object
  DriveServer ds = new()
  ds.ClientID = "fomkd5ca0txacm3"
  ds.ClientSecret = "vf2z9jxs0kvn981"
  ds.AccessToken = "it8u8hh3xp5xjdr"
  ds.SecretToken = "ohdsvbn7n8vfx8a"
  ds.ServerType = Dropbox
  ds.Idform = this.IDForm()
  //
  // connect to the server
  ds.Connect(true, false)
}
```

7.7.3 Lista dei file e delle directory

Per ottenere la lista dei file e delle directory di primo livello presenti sul server è necessario utilizzare la procedura *Load Root Items* che ha un parametro opzionale intero che

può essere utilizzato per indicare quanti livelli si vogliono caricare. Se viene lasciato vuoto verranno caricati tutti i livelli.

Se invece si vuole ottenere la lista dei file e delle directory di una cartella diversa dalla radice, occorre utilizzare la procedure *Load Child Items* che ha un parametro opzionale intero che può essere utilizzato per indicare quanti livelli si vogliono caricare. Se viene lasciato vuoto verranno caricati tutti i livelli.

Se, ad esempio, si vogliono ottenere tutti i file e le directory di primo livello occorre scrivere il seguente codice:

```
public void Form.DropboxLoadItems()
{
  // create a Drive Server object
  DriveServer ds = new()
  ds.ClientID = "fomkd5ca0txacm3"
  ds.ClientSecret = "vf2z9jxs0kvn981"
  ds.AccessToken = "it8u8hh3xp5xjdr"
  ds.SecretToken = "ohdsvbn7n8vfx8a"
  ds.ServerType = Dropbox
  ds.Idform = this.IDForm()
  //
  // connect to the server
  ds.Connect(true, this.IDForm(), false)
  //
  if (ds.isConnected())
  {
    // load first level files and directories
    ds.LoadRootItems(1)
  }
}
```

Se è la prima volta che ci si connette, la chiamata alla funzione *isConnected* restituirà *false* perché è necessario prima ottenere il Secret Token e l'autorizzazione dell'utente. L'utente sarà quindi redirezionato sul sito di Dropbox dove dovrà accedere con il suo username e la sua password e poi autorizzare l'applicazione. Se l'operazione va a buon fine, l'utente sarà redirezionato di nuovo sull'applicazione e l'applicazione invierà un messaggio "OAUTH" all'Idform. Per intercettare il messaggio occorre scrivere il seguente codice:

```
event Form.OnSendMessage(
   string Message // Indicates the name of the message
   IDForm Sender  // Identifies the form that sent the message. IDForm type...
   IDDocument Doc // Optional document to be sent to the sub-form
   string Par1    //
   string Par2    //
   string Par3    //
   string Par4    //
)
{
   if (Message == "OK")
   {
      DriveServer ds = (DriveServer)Doc
      ds.LoadRootItems(1)
   }
}
```

7.7.4 Upload di un file

Per caricare un file su Dropbox occorre creare un oggetto di tipo *Server Item* e impostare le seguenti proprietà:

Name	Nome del file
Path	Path del file
Type	Vale *File* nel caso in cui si stia caricando un file, *Directory* se si sta creando una nuova directory

Occorre poi aggiungerlo alla collection che rappresenta la cartella in cui deve essere inserito. Dopo aver creato in questo modo tutti gli oggetti da caricare, occorre salvare le collection in cui sono stati aggiunti.

Se, ad esempio, si vuole caricare il file *report.pdf* nella cartella *Documents* è possibile scrivere il codice mostrato alla pagina seguente.

```
public void Form.DropboxUploadFile()
{
  // create a Drive Server object
  DriveServer ds = new()
  ds.ClientID = "fomkd5ca0txacm3"
  ds.ClientSecret = "vf2z9jxs0kvn981"
  ds.AccessToken = "it8u8hh3xp5xjdr"
  ds.SecretToken = "ohdsvbn7n8vfx8a"
  ds.ServerType = Dropbox
  ds.Idform = this.IDForm()
  //
  // connect to the server
  ds.Connect(true, this.IDForm(), false)
  //
  if (ds.isConnected())
  {
    // load first level files and directories
    ds.LoadRootItems(1)
    //
    // search "Documents" directory
    ServerItem rootDir = ds.RootDirectory
    for each ServerItem si in rootDir.ServerItems
    {
      if (si.Type == Directory && si.Name == "Documents")
      {
        // create a Server Item object
        ServerItem fileToLoad = new()
        fileToLoad.init()
        fileToLoad.Name = "report.pdf"
        fileToLoad.Path = FileExplorer.path() + "/temp"
        fileToLoad.Type = File
        //
        // add the item to the collection
        si.ServerItems.add(fileToLoad)
        //
        // save the collection
        si.ServerItems.saveToDB([childrenlevel], [revalidate])
      }
    }
  }
}
```

7.7.5 Download di un file

Per scaricare un file da Dropbox occorre usare la procedura *Download* dell'oggetto *Server Item*. Questa procedura ha un parametro di tipo stringa in cui specificare il percorso assoluto, comprensivo del nome del file, dove verrà scaricato il file e un secondo parametro che non viene utilizzato da Dropbox.

Se, ad esempio, si vuole scaricare il file *report.pdf* nella cartella *temp* dell'applicazione occorre scrivere il seguente codice:

```
public void Form.DropboxDownloadFile()
{
  // create a Drive Server object
  DriveServer ds = new()
  ds.ClientID = "fomkd5ca0txacm3"
  ds.ClientSecret = "vf2z9jxs0kvn981"
  ds.AccessToken = "it8u8hh3xp5xjdr"
  ds.SecretToken = "ohdsvbn7n8vfx8a"
  ds.ServerType = Dropbox
  ds.Idform = this.IDForm()
  //
  // connect to the server
  ds.Connect(true, this.IDForm(), false)
  //
  if (ds.isConnected())
  {
    // load first level files and directories
    ds.LoadRootItems(1)
    //
    // search "Documents" directory
    ServerItem rootDir = ds.RootDirectory
    for each ServerItem si in rootDir.ServerItems
    {
      if (si.Type == Directory && si.Name == "Documents")
      {
        // load "Documents" children
        si.LoadChildItems(1)
        //
        // search "report.pdf"
        for each ServerItem file in si.ServerItems
        {
          if (file.Type == File && file.Name == "report.pdf")
          {
            file.Download(FileExplorer.path() + "/temp/" + file.Name,
                [format])
          }
        }
      }
    }
  }
}
```

7.7.6 Eliminazione di file e cartelle

Per cancellare un file o una cartella occorre marcarlo per la cancellazione impostando a true la proprietà *Deleted*. Dopo aver eliminato tutti gli elementi che si desiderava cancellare occorre salvare la collection (o le collection) che li contiene.

Se, ad esempio, si vuole cancellare il file *report.pdf* contenuto nella directory *Documents* occorre scrivere il seguente codice:

```
public void Form.DropboxDeleteFile()
{
  // create a Drive Server object
  DriveServer ds = new()
  ds.ClientID = "fomkd5ca0txacm3"
  ds.ClientSecret = "vf2z9jxs0kvn981"
  ds.AccessToken = "it8u8hh3xp5xjdr"
  ds.SecretToken = "ohdsvbn7n8vfx8a"
  ds.ServerType = Dropbox
  ds.Idform = this.IDForm()
  //
  // connect to the server
  ds.Connect(true, this.IDForm(), false)
  //
  if (ds.isConnected())
  {
    // load first level files and directories
    ds.LoadRootItems(1)
    //
    // search "Documents" directory
    ServerItem rootDir = ds.RootDirectory
    for each ServerItem si in rootDir.ServerItems
    {
      if (si.Type == Directory && si.Name == "Documents")
      {
        // load "Documents" children
        si.LoadChildItems(1)
        //
        // search "report.pdf"
        for each ServerItem file in si.ServerItems
        {
          if (file.Type == File && file.Name == "report.pdf")
          {
            file.deleted = true
            //
            // save the collection
            si.ServerItems.saveToDB([childrenlevel], [revalidate])
            //
            break
          }
        }
        //
        break
      }
    }
  }
}
```

7.7.7 Copia e spostamento di file

Per copiare un file da una cartella a un'altra occorre usare la funzione *CopyTo* che ha un parametro *Server Item* da usare per specificare la destinazione e un secondo parametro opzionale da usare se si vuole cambiare il nome del file copiato. La funzione restituisce l'oggetto copiato.

Se, ad esempio, si vuole copiare il file *report.pdf* nella cartella *Public*, occorre scrivere il seguente codice:

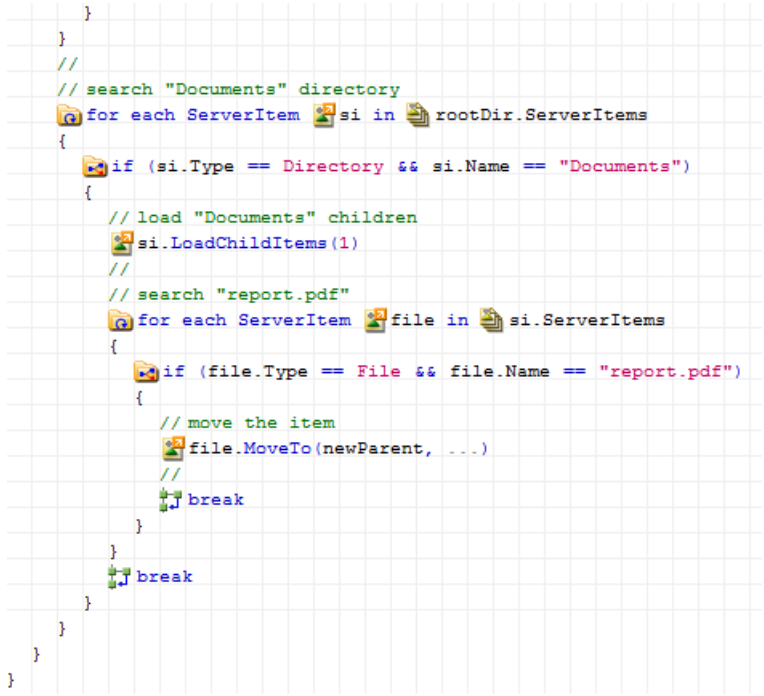
```
public void Form.DropboxCopyFile()
{
  // create a Drive Server object
  DriveServer ds = new()
  ds.ClientID = "fomkd5ca0txacm3"
  ds.ClientSecret = "vf2z9jxs0kvn981"
  ds.AccessToken = "it8u8hh3xp5xjdr"
  ds.SecretToken = "ohdsvbn7n8vfx8a"
  ds.ServerType = Dropbox
  ds.Idform = this.IDForm()
  //
  // connect to the server
  ds.Connect(true, false)
  //
  if (ds.isConnected())
  {
    // load first level files and directories
    ds.LoadRootItems(1)
    //
    ServerItem rootDir = ds.RootDirectory
    //
    // search "Public" directory
    ServerItem newParent = new()
    for each ServerItem si in rootDir.ServerItems
    {
      if (si.Type == Directory && si.Name == "Public")
      {
        newParent = si
        //
        break
      }
    }
    //
    // search "Documents" directory
    for each ServerItem si in rootDir.ServerItems
    {
      if (si.Type == Directory && si.Name == "Documents")
      {
        // load "Documents" children
        si.LoadChildItems(1)
        //
        // search "report.pdf"
```

```
            for each ServerItem file in si.ServerItems
            {
               if (file.Type == File && file.Name == "report.pdf")
               {
                  // copy the item
                  file.CopyTo(newParent, ...)
                  //
                  break
               }
            }
            break
         }
      }
   }
}
```

Per spostare un file da una cartella a un'altra occorre usare la funzione *MoveTo* che ha un parametro *Server Item* da usare per specificare la destinazione e un secondo parametro opzionale da usare se si vuole cambiare il nome del file spostato. La funzione restituisce l'oggetto spostato.

Se, ad esempio, si vuole spostare il file *report.pdf* dalla sua cartella *Documents* alla cartella *Public*, occorre scrivere il seguente codice:

```
public void Form.DropboxMoveFile()
{
   // create a Drive Server object
   DriveServer ds = new()
   ds.ClientID = "fomkd5ca0txacm3"
   ds.ClientSecret = "vf2z9jxs0kvn981"
   ds.AccessToken = "it8u8hh3xp5xjdr"
   ds.SecretToken = "ohdsvbn7n8vfx8a"
   ds.ServerType = Dropbox
   ds.Idform = this.IDForm()
   //
   // connect to the server
   ds.Connect(true, false)
   //
   if (ds.isConnected())
   {
      // load first level files and directories
      ds.LoadRootItems(1)
      //
      ServerItem rootDir = ds.RootDirectory
      //
      // search "Public" directory
      ServerItem newParent = new()
      for each ServerItem si in rootDir.ServerItems
      {
         if (si.Type == Directory && si.Name == "Public")
         {
            newParent = si
            //
            break
```



```
        }
    }
    //
    // search "Documents" directory
    for each ServerItem si in rootDir.ServerItems
    {
        if (si.Type == Directory && si.Name == "Documents")
        {
            // load "Documents" children
            si.LoadChildItems(1)
            //
            // search "report.pdf"
            for each ServerItem file in si.ServerItems
            {
                if (file.Type == File && file.Name == "report.pdf")
                {
                    // move the item
                    file.MoveTo(newParent, ...)
                    //
                    break
                }
            }
            break
        }
    }
}
}
```

7.7.8 Test del servizio Dropbox

Per provare il servizio Dropbox tramite l'applicazione di esempio IDCloud è necessario selezionare Dropbox nella schermata iniziale e poi scrivere le credenziali nei rispettivi campi. Cliccando sul bottone con la freccia verrà visualizzata la videata che permette di navigare tra le directory e i file su Dropbox dell'utente.

Nella colonna di sinistra sono mostrate le directory. Cliccando su una di esse verrà mostrata, sulla destra, la lista dei file che contiene. Per aggiungere una nuova directory occorre cliccare sul bottone con la stella, scrivere il nome della nuova directory e premere OK**.** Per eliminare una directory occorre cliccarci sopra e poi cliccare il bottone con la x rossa.

Per aggiungere un nuovo file occorre premere il bottone con la stella presente sopra la lista dei file e indicare qual è il file che si vuole caricare. Per eliminare un file occorre premere il pulsante con la x rossa presente nella stessa riga del file da cancellare. Per scaricare un file occorre premere il pulsante con la freccia verde presente sulla stessa riga del file da scaricare.

7.8 Servizio Google Drive

Il servizio *Google Drive* permette di gestire i file di relativi all'interno delle applicazioni realizzate con Instant Developer. Le funzionalità del servizio sono le seguenti:

- creazione cartelle;
- lista dei file e directory presenti sul server;
- upload, download di file;
- eliminazione di file e cartelle;
- copia e spostamento di file da una cartella a un'altra.

Per poter utilizzare il servizio è necessario ottenere le credenziali registrando una nuova applicazione al sito https://code.google.com/apis/console. Se si sta registrando un'applicazione per la prima volta, si presenterà la seguente videata:

Videata iniziale console sviluppatori di Google

In caso contrario si aprirà direttamente la console per la gestione delle applicazioni. Per registrare una nuova applicazione occorre compiere le seguenti azioni:

- Se è la prima volta che si accede alla console sviluppatori, premere il pulsante *Create project…*
- Se non è la prima volta, cliccare sul menu a tendina *API Project* e scegliere *Create*. Nella finestra che si aprirà, inserire il nome del progetto e premere il pulsante *Create project.*

- Scegliere *Services* dal menu sulla sinistra e abilitare le *Drive API* cliccando sul pulsante OFF e accettando poi i termini del servizio.
- Scegliere *API Access* dal menu sulla sinistra e premere il pulsante *Create an OAuth 2.0 client ID…*
- Nella finestra che si aprirà, scrivere almeno il *Product name* cliccare su *Next.*
- Nella finestra successiva scegliere *Web Application* come tipo di applicazione.
- Cliccare su *more option* e scrivere nello spazio relativo agli *Authorized Redirect URIs* l'URL autorizzato al redirect. Questo URL si ottiene da quello dell'applicazione sostituendo il nome della pagina con *IDCloud.htm?CMD=AUTH*. Ad esempio, se l'URL dell'applicazione è *http://www.progamma.com/Applicazione/Applicazione.aspx*, l'URL da scrivere sarà *http://www.progamma.com/-Applicazione/IDCloud.htm?AUTH*. Può essere specificato più di un URL.
- Premere il bottone *Create client ID* per ottenere le credenziali d'accesso.
- Le credenziali d'accesso sono indicate nella pagina *API Access* nell'area *Client ID for web applications.*

7.8.1 Connessione al server

Per poter effettuare qualsiasi operazione su file e cartelle occorre prima essere connessi al server Google. Per connettersi è necessario creare un oggetto di tipo *Drive Server* e impostare le seguenti proprietà:

Client ID	Corrisponde all'App key assegnato in seguito alla registrazione dell'applicazione sul sito sviluppatori di Google.
Client Secret	Corrisponde all'App secret assegnato in seguito alla registrazione dell'applicazione sul sito sviluppatori di Google.
Access Token	Valore utilizzato per permettere all'applicazione di accedere alle risorse per conto dell'utente. Se si possiede già un access token valido lo si può indicare usando questo parametro.
Refresh Token	Valore utilizzato per ottenere un nuovo Access Token quando quest'ultimo scade. Se si possiede già un refresh token valido lo si può indicare usando questo parametro.
Server Type	Indica a quale serve ci si vuole connettere. I valori possibili sono: Dropbox, GoogleDrive e SkyDrive
Idform	Id della form che riceverà i messaggi inviati dal componente, se presente.

Occorre quindi usare il metodo *Connect* che ha un parametro booleano con il quale deve essere specificato se l'applicazione ha un'interfaccia grafica e un altro parametro booleano che deve essere impostato a *true* se si desidera l'accesso in sola lettura. Il metodo *Connect* redireziona l'utente sulla pagina di Google in cui dovrà fare login e poi, la prima volta che si connette, autorizzare Google ad accedere ai suoi dati.

Se la connessione al server non va a buon fine verrà inviato un messaggio "Authorization Error" o "Verification Error" all'Idform e verrà impostato il parametro *Last Error Message*.

Se la connessione al server va a buon fine verrà inviato un messaggio "OAUTH" all'Idform, verrà impostata la *Root Directory* e le proprietà *User Name* e *User ID* dell'oggetto *Drive Server*.

L'Access Token ha un tempo di vita limitato indicato dalla proprietà *Expires In* dell'oggetto *Drive Server*. Tuttavia l'oggetto *Drive Server* è in grado di recuperare in modo automatico un nuovo Access Token quando quello vecchio scade preoccupandosi quindi di mantenere attiva la connessione.

Un esempio di codice è il seguente:

```
public void Form.GDriveConnect()
{
  // create a Drive Server object
  DriveServer ds = new()
  ds.ClientID = "254986321475.apps.googleusercontent.com"
  ds.ClientSecret = "KODFsYJicvKDRljOfUhvnICU"
  ds.AccessToken = "ya29.AHES6ZSGR7J4MnFEGgIXdV6LYLkWnVeEi5fBIiSC_zIj3fG3Uoiuww"
  ds.RefreshToken = "1/YLFJibcBE-Z0uhI_lkTDJzDF1K-2YKZ5DL1AXpHrpdl"
  ds.ServerType = GoogleDrive
  ds.Idform = this.IDForm()
  //
  // connect to the server
  ds.Connect(true, false)
}
```

Per tutte le altre operazioni, si possono seguire le procedure descritte per Dropbox con le seguenti differenze:

- le procedure *Load Root Items* e *Load Child Items* non utilizzano il parametro *levels* ma caricano sempre tutti i livelli
- quando si carica un file su Google Drive si può specificare anche la proprietà *Convert* che può essere impostata a *true* se si vuole convertire il file nel corrispondente formato Google Docs.
- l'Access Token ha una durata di 3600 secondi circa, indicata dal parametro *ExpiresIn*. Il refresh dell'Access Token avviene in automatico e quando si verifica verrà inviato un messaggio "TOKENREFRESHED" all'Idform indicata nel parametro *Idform* dell'oggetto *Drive Server*.

7.9 SkyDrive

Il servizio SkyDrive permette di gestire i file su SkyDrive all'interno delle applicazioni realizzate con Instant Developer. Le funzionalità del servizio sono le seguenti:

- creazione cartelle
- lista dei file e directory presenti sul server
- upload, download di file
- eliminazione di file e cartelle
- copia e spostamento di file da una cartella a un'altra

Per poter utilizzare il servizio è necessario ottenere le credenziali registrando una nuova applicazione sul sito sviluppatori di Microsoft.

Per registrare un'applicazione occorre cliccare su *My apps* nel menu in alto. Nella nuova pagina, cliccare su *Create application*, specificare poi il nome dell'applicazione, la lingua e accettare i termini di utilizzo. Nella pagina successiva ci sarà il riepilogo delle credenziali e sarà necessario indicare il dominio dell'applicazione e il tipo, per il quale occorre selezionare *No*. Premere *Save* per salvare la nuova applicazione.

7.9.1 Connessione al server

Per poter effettuare qualsiasi operazione su file e cartelle occorre prima essere connessi al server. Per connettersi è necessario creare un oggetto di tipo *Drive Server* e impostare le seguenti proprietà:

Client ID	Corrisponde all'App key assegnato in seguito alla registrazione dell'applicazione sul sito sviluppatori di Dropbox.
Client Secret	Corrisponde all'App secret assegnato in seguito alla registrazione dell'applicazione sul sito sviluppatori di Dropbox.
Access Token	Valore utilizzato per permettere all'applicazione di accedere alle risorse per conto dell'utente. Se si possiede già un access token valido lo si può indicare usando questo parametro.
Refresh Token	Valore utilizzato per ottenere un nuovo Access Token quando quest'ultimo scade. Se si possiede già un refresh token valido lo si può indicare usando questo parametro.
Server Type	Indica a quale serve ci si vuole connettere. I valori possibili sono: Dropbox, GoogleDrive e SkyDrive
Idform	Id della form che riceverà i messaggi inviati dal componente, se presente.

Occorre quindi usare il metodo *Connect* che ha un parametro booleano con il quale deve essere specificato se l'applicazione ha un'interfaccia grafica e un altro parametro booleano che deve essere impostato a *true* se si desidera l'accesso in sola lettura. Il metodo *Connect* redireziona l'utente sulla pagina di Microsoft in cui dovrà fare login e poi, la prima volta che si connette, autorizzare SkyDrive ad accedere ai suoi dati.

Se la connessione al server non va a buon fine verrà inviato un messaggio "Authorization Error" o "Verification Error" all'Idform e verrà impostato il parametro *Last Error Message*.

Se la connessione al server va a buon fine verrà inviato un messaggio "OAUTH" all'Idform, verrà impostata la *Root Directory* e le proprietà *User Name* e *User ID* dell'oggetto *Drive Server*.

L'Access Token ha un tempo di vita limitato indicato dalla proprietà *Expires In* dell'oggetto *Drive Server*. Tuttavia l'oggetto *Drive Server* è in grado di recuperare in modo automatico un nuovo Access Token quando quello vecchio scade preoccupandosi quindi di mantenere attiva la connessione.

Se, ad esempio, si vuole accedere al server in lettura e scrittura occorre scrivere il seguente codice:

```
public void Form.SkyDriveConnect()
{
  // create a Drive Server object
  DriveServer ds = new()
  ds.ClientID = "000000000539J984Z"
  ds.ClientSecret = "a5mWlekfMWxEdfVukUDlSo9fVp2Sb32u"
  ds.AccessToken = "EwBAAq1DBAAUlbRWyAJjK5w968Ru3Cyt/6GvwXwAAXZ5BI+WsUJajshIW3X-
          lXLpqaMFGB/d-8Eer87O8S1dKeIMoA15kqeqjKyUD9w6dQ0WEfsB3fgY7mnFBgt17kFcJ83-
          BNxYElkiqVIgaTrHHSIRvoxE-3jjeYkHKvcipAZk+PTGmQjHMSW0xHrYFEVVgoNdp3eDlYq-
          BB6f1ZEQOfYngb6+6Txav6F+h3zckOsJmjrk-CVe9G217E9JfhD0jrj/Foed8Q7GtTpwGCM-
          3K0GFtdsTITzushvU/puwP7rhCakOveM8lNUIa9IbvRq1GQ8-yiXgmC/o7bWfa9WT3gopCx-
          l6U+pO0mCYRIvyrtT4PpMJhAwiB3HMr24dLZmdr3Bk2cDZgAACNLr1p0o4G9-ZEAHf5nsvf-
          RwGNrHgdzndCp9XJ5HDQ/9RvtPjTrhPeucSwh9CQZ0miAJyIlkJlwTkPreYdDh7rXYcbx/j-
          d-ug5pWRT+kthzTRy+78Zr0XXMbOi+pYNncvvbQ/6yz+CatEyXBFFca/y0t24kB52enyjN/-
          ieeWib+qWHN3L-gUheTZHFLTqb2rvdj2Vs6Wtz4XmSOnIWORblS4LxfveeOafYTJACMDRbO-
          OOHSu5g/yl9XEDVej8DLTmRyJ-C4FJvzZzP7bEWwLcz4B9F8YrwhYJtOJ9Kgo7bR/98vLPw-
          /7B64IaFr0w/IRK55Y/tijrjkoDnULQcZCm4d-8gH7HL11N/4kSKiyEW1LrtZQWcvCIui8N-
          kBvVuwAA"
  ds.RefreshToken = "ChQTmD3v9KjczWfUen9wJS!2L7bukZE86n1jXF3cZEsWCHB!cZ0e1XZKHD-
          mxqfDaSLErM6-LLBAXWq*8Hoag2qN54haJohA6I0oLH0S2hUMGiS45I7O3I3ByZojqD*kmp-
          GYkYIlgbZlZA0Y1gI*FhDXua-VMFv1c8jSC8ys!*jZuJ1bVuq7P5!VoDYASsCS5Td2ME47J-
          8E6*WgVhuv1oAGOItCJ0jY9qHabmOLjlgTOS-uCRRGmdw!T9KXdRP0AcmXFkZRRB8n4e!E!-
          3zL51vt*!jq7diM!44lhZpP8sjK1EVBoycOx1PM1so7Ck4ol-Tr2ojIvjg4seWoJAO4j5Sd-
          n!UkIowmwpp8aymuHKPWWLcVj1PhMUWqbC3WZhRj*VAWr7jVfvbgHstdFr1i-KJnH51u*fR-
          EYF05v5tW1!IiMOOqdlv"
  ds.ServerType = SkyDrive
  ds.Idform = this.IDForm()
  //
  // connect to the server
  ds.Connect(true, false)
}
```

Per tutte le altre operazioni, si possono seguire le procedure descritte per Dropbox con la seguente differenza:

- l'Access Token ha una durata di 3600 secondi circa, indicata dal parametro *ExpiresIn*. Il refresh dell'Access Token avviene in automatico e quando si verifica verrà inviato un messaggio "TOKENREFRESHED" all'Idform indicata nel parametro *Idform* dell'oggetto *Drive Server*.

7.10 Applicazioni mobile offline

Per poter tornare indietro correttamente dopo un redirect verso una pagina esterna, l'applicazione usa l'URL memorizzato nella proprietà globale *Callback URL*. Il suo valore viene calcolato in automatico dal componente nell'evento *Initialize* ed è diverso nel caso di applicazioni online o offline.

Se l'applicazione è online il callback URL è l'URL dell'applicazione stessa. Se invece si tratta di un'applicazione mobile offline eseguita dentro la shell nativa, sarà l'URL del file IDCloud.htm. Tale URL è calcolato dall'applicazione nei seguenti modi:

- se è stato configurato un server di sincronizzazione dentro alla pagina di configurazione dei servizi nel sito CRMID, Caravel utilizza tale URL sostituendo il nome della pagina web dell'applicazione con il file IDCloud.htm;
- se l'applicazione non conosce l'URL del server di sincronizzazione, viene utilizzato l'URL del sito da cui viene scaricata l'applicazione offline.

In alcuni casi non è possibile utilizzare il callback URL calcolato automaticamente ma è necessario specificare l'URL del file IDCloud.htm valorizzando la proprietà globale *CallbackURL* del componente nell'evento *Initialize* dell'applicazione:

- quando l'applicazione che si sta sviluppando deve funzionare su Windows 8 e non viene configurato un server di sincronizzazione;
- quando si vuole creare un pacchetto di installazione per l'applicazione;
- quando si vuole memorizzare il file IDCloud.htm in una posizione diversa rispetto a quella calcolata dal componente.