

Capitolo 6

Componente IDCloud

6.1 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.

6.2 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.

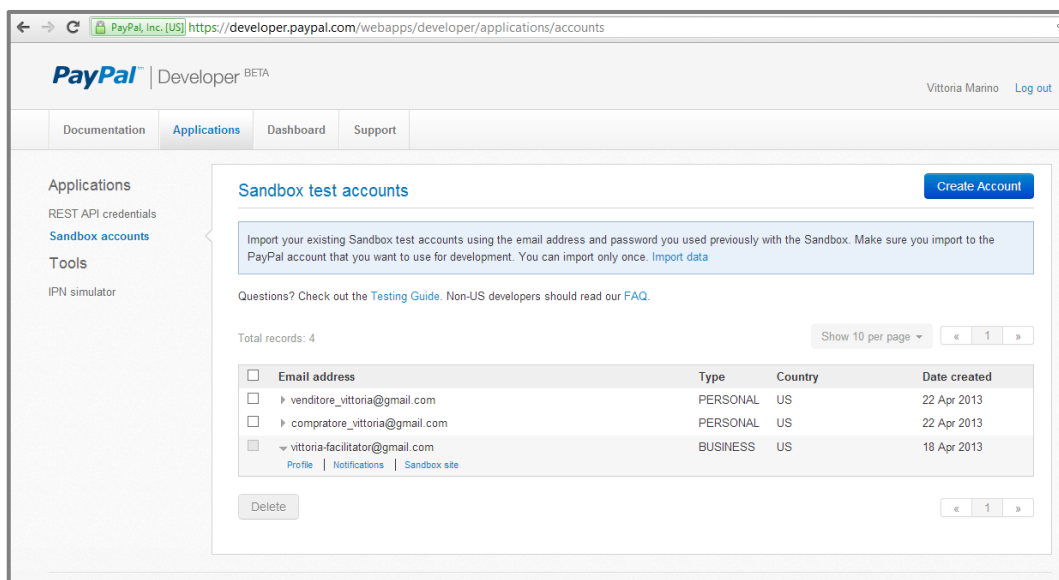
Per poterlo utilizzare è necessario essere in possesso delle credenziali. Si possono richiedere due tipi di credenziali: credenziali di test e credenziali di produzione.

6.2.1 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.

6.2.2 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.
- Effettuare 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.

6.2.3 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 <i>RecurringPayment</i> . In tutti gli altri casi deve essere lasciata vuota.

Componente IDCloud

Billing Agreement Description	Descrizione dei beni o servizi associati al tipo di pagamento impostato nel parametro Billing Type.
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.

The screenshot displays the PayPal checkout interface. On the left, a 'Your order summary' box lists items: 'Tablet case' (€40.00) and 'Bluetooth headphones' (€40.00). The total is €80.00, with a tax of €30.00, resulting in a final total of €110.00 EUR. On the right, under 'Choose a way to pay', the 'Pay with my PayPal account' option is selected. It shows a login form with email 'buyer@mail.com' and a masked password. A 'Log In' button is present, along with a link to 'Forgot your email address or password?'. Below this, there is a section for 'Don't have a PayPal account?' with an optional link to join for faster checkout. At the bottom, a link to 'Cancel and return to Venditore Test's Test Store.' is visible. The footer includes a 'Site Feedback' link and a note about PayPal's safety and links to the 'User Agreement' and 'Privacy Policy'.

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.

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_apil@gmail.com"
    ps.Signature = "A5zZOi.7c9ngcAOVmf7hjJbH9I4AZi-XT1DjebHFK1SgSCh2xbzjkV3"
    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)
}
```

6.2.4 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_apil@gmail.com"
    ps.Signature = "A5zZOi.7c9ngcAOVmfd7hjJbH9I4AZi-XT1DjebHfKl8qSCH2xbzjkV3"
    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)
}
```

6.2.5 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_apil@gmail.com"
    ps.Password = "652115677"
    ps.Signature = "A5zZOi.7c9ngcAOVmfd7hjJbH9I4AZi-XT1DjebHfKl8qSCH2xbzjkV3"
    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
}
```

6.2.6 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@gmail.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

Available Balance 1723.24 EUR

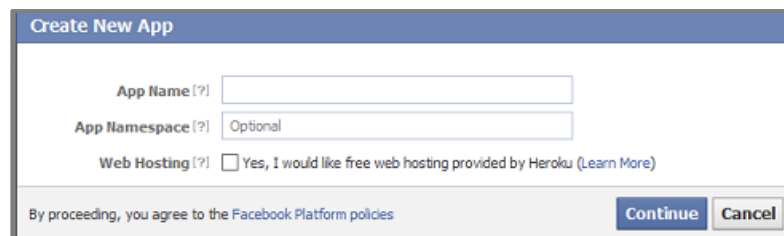
Cliccando poi sul bottone *Pay with PayPal* verrà simulato il pagamento di un ordine. Si verrà redirezionati sul sito di test di PayPal dove sarà necessario fare il login con le credenziali di uno degli account di prova impostati nel proprio account PayPal sviluppatore. Se si conferma il pagamento, al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

6.3 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.

6.3.1 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 = "CAACGZCju7wm8BAIS86bM1pinIOhFBPOMzBiRcyZAI81UKZAxbsURdB-
        XKDqD1FwNiARxJ2FtJ9rFM1WjL8QkbMD0dHak9ZBYfXKdLMYzBVWJALRlor4vtz4Kb-
        yq8rmqSIch7cLfGZBkHarFlj6B32R8ZADkTcaVWQgkZD"
    //
    fs.Connect(true, ReadOnly + WriteOnly)
}
```

6.3.2 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)
    }
}
```

6.3.3 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-
    ERH1AtfZC1kKczZBWJ5v3YQ1UhD5dKN1ZCXnZBjdRswD6D76ZAFje9YzHqqlvXc4i7-
    WZCCZCZCOX3gWgHM4rlxiYVfdoYkeyDoD2AYliyW0s8kZD"

    //
    // connect to the server
    fs.Connect(true, ReadOnly)
    //
    // load status items
    fs.loadCollectionFromDB(fs.StatusItems, ...)
}
```

6.3.4 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-
    0xaEiwQACZAQX6hHmf4qCt1x2ClafIhAZAYL7ZBT2AyNNCHV4ZD"

    //
    // connect to the server
    fs.Connect(true, ReadOnly)
    //
    fs.loadCollectionFromDB(fs.Photos, ...)
}
```

6.3.5 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:

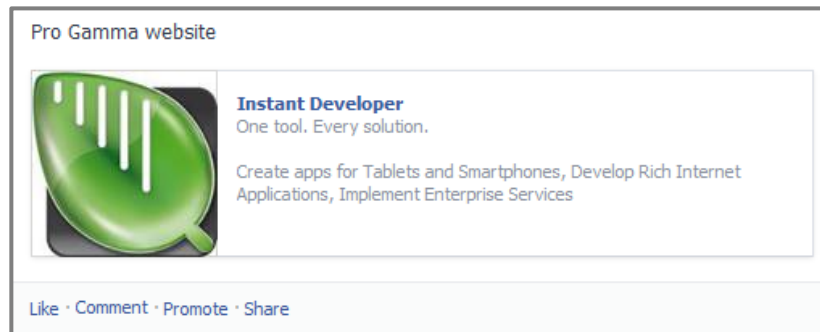
```
public void Form.PostFacebook()
{
    // create a Facebook Server object
    FacebookServer fs = new()
    fs.ClientID = "985216654233254"
    fs.ClientSecret = "b354462ef2c1f319b0991ca82a412253"
    fs.AccessToken = "CAACGZCju7wm8BAB2WeMt7g1NoJoOZAKv5nQbK5TdbRP301ESsPZBge-
        OndCDgHbsJa42CdZB21DtpLC6bJxqFGwxIAThC6D90Btebo6YkamnbAa0rBh7j41xi-
        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:



6.3.6 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 = "CAACGZCju7wm8BAGcnuEFap1Mt723TrIUJbIw7aidcTlOmJqgeJIpIu-
Pfcn0TcMVKDGhzbMGt3V8U87RNhO56O7nZB8JDWGTf2BLloPD2A7WZBEfzAAqt113k-
HivL4ItWo1J8ch7mRkThgV3ywwmVS1x8qogodJUYZD"
    //
    // 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(...)
}

```

6.3.7 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:

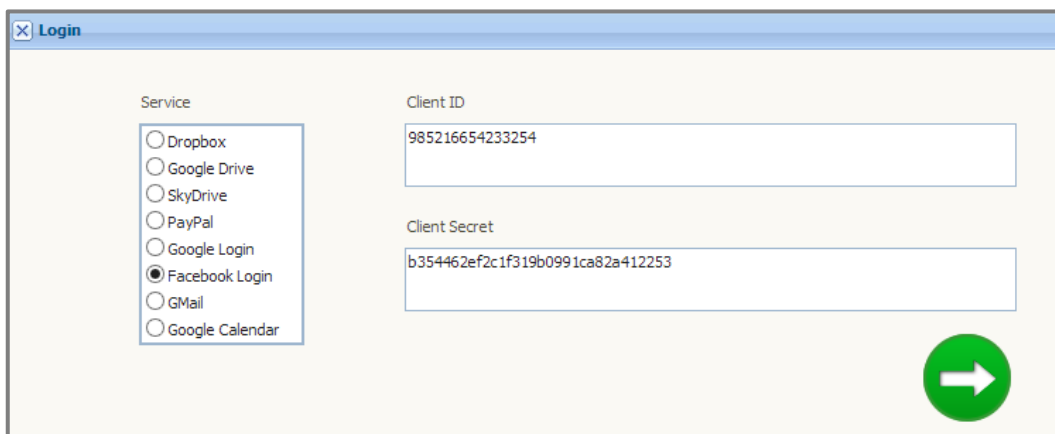
ID	Identificatore univoco del commento.
From	L'utente che ha scritto il commento.
Message	Il testo del commento.
Created Time	La data di pubblicazione del commento.
Like Count	Il numero di like del commento.

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-  
        ERH1AtfZC1kKczZBWJ5v3YQ1UhD5dKN1ZCXnZBjdRswD6D76ZAFje9YzHqqlvXc4i7-  
        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  
        }  
    }  
}
```

6.3.8 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.

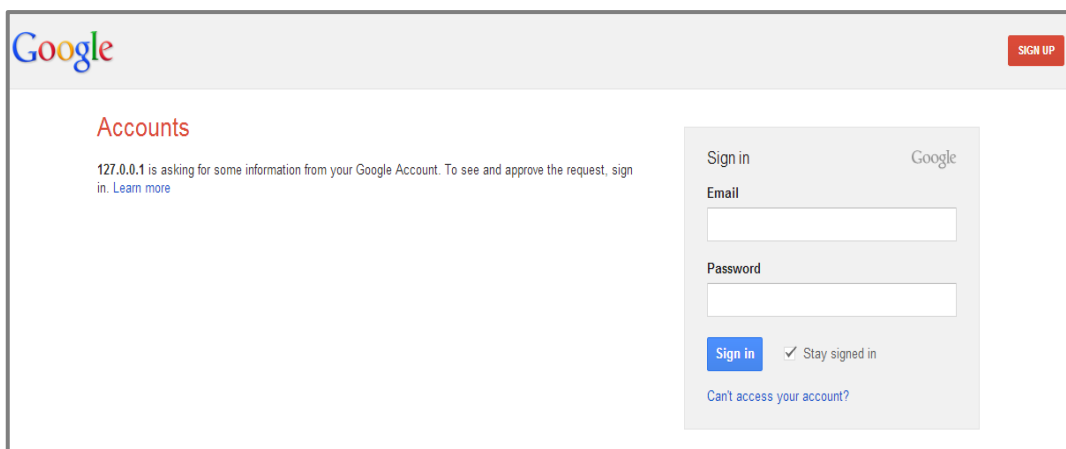


Cliccando sul bottone con la freccia verrà visualizzata la videata che permette di effettuare il login tramite Facebook. Questa videata contiene un solo pulsante *Login with Facebook* che quando premuto reindirizza il browser alla pagina di login di Facebook dove sarà necessario inserire prima email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

6.4 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".

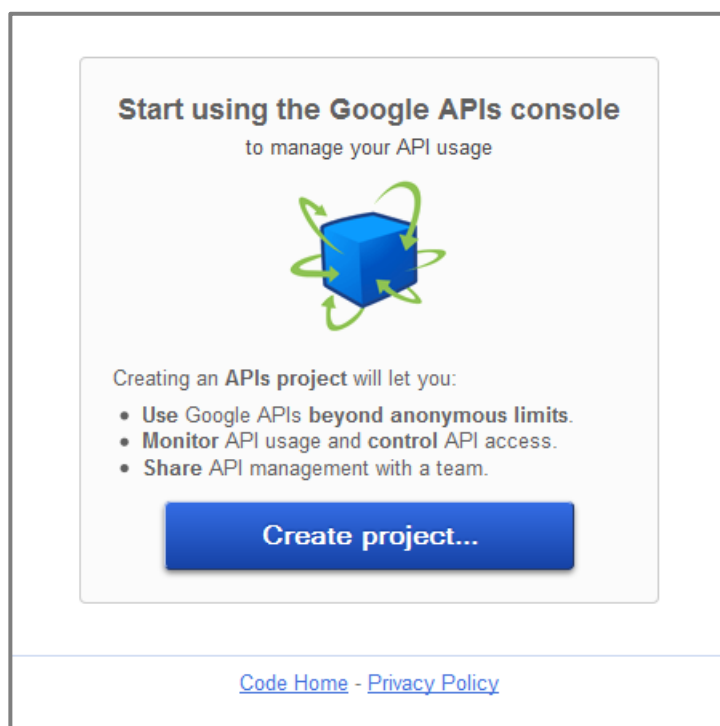
Premendo questo bottone si verrà reindirizzati alla pagina di login di Google dove sarà necessario inserire prima email e password e poi autorizzare l'accesso ai dati dell'account e alla mail. Dopo aver cliccato su "Accetto" si verrà reindirizzati ad una videata dell'applicazione che riepiloga i dati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

6.5 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*.

6.5.1 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.

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 = "JEPDvERicvUODnuOfUnedOPG"
    cs.AccessToken = "ya29.AHES6ZRT17Ui1MLtTA1VY4E8XDOgFyoMHvx12R1x1iVhcVrCfGmaYp0M"
    cs.RefreshToken = "1/Pyn2DF_sGmlZ_zYnmQQaQxK0di6ZpUPEn6yyLu3VEaA"
    //
    // connection to the server
    cs.Connect(true, false)
}
```

6.5.2 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.AHES6ZRT17Ui1MLtTA1VY4E8XDOgFyoMHvx12R1x1iVhcVrCfGmaYp0M"
    cs.RefreshToken = "1/Pyn2DF_sGmlZ_zYnmQqaQxK0di6ZpUPEn6yyLu3VEaA"
    //
    // connection to the server
    cs.Connect(true, false)
    //
    // load calendars
    cs.loadCollectionFromDB(cs.Calendars, [childrenlevel])
}

```

6.5.3 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.

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 istanze 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.AHES6ZRT17Ui1MLtTA1VY4ESXDOgFyoMHvx12R1x1iVhcVrCfismaYp0M"
    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
    }
}
```

6.5.4 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.AHES6ZRT17UilMLtTA1VY4E8XDOgFyoMHvx12R1x1iVhcVrCfGmaYp0M"
    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
}
}
```

6.5.5 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 h104m268294uekunrqbpln9pvs_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 = "JEPDvERicvUODnuOfUnedOPG"
    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 == "h104m268294uekunrqbpln9pvs_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
}
}
```

6.5.6 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 `h104m268294uekunrqbpln9pvs_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_zYnmQqQxK0di6ZpUPEn6yyLu3VEaA"
    //
    // 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 == "h104m268294uekunrqbpln9pvs_20130218T110000Z")
                {

```

```
// delete the event
ei.deleted = true
ci.Events.saveToDB(...)
//
break
}
//
break
}
}
```

6.5.7 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.

Service

- ☐ Dropbox
- ☐ Google Drive
- ☐ SkyDrive
- ☐ PayPal
- ☐ Google Login
- ☐ Facebook Login
- ☐ Gmail
- ☒ Google Calendar

Client ID

365482156985.apps.googleusercontent.com

Client Secret

JEPDvERicvUODnuOfUncdOPG

Access Token

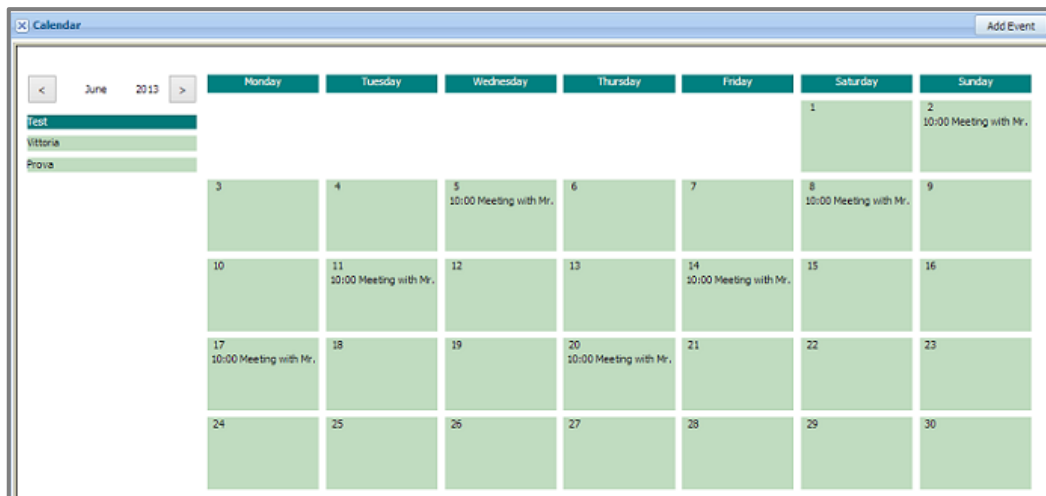
ya29.AHES6ZRI3vm_tvG8bVYrLsPPg0afJANSLObNDXyyR_dyRIJ6dqMb7M

Refresh Token

1/Pyn2DF_sGmIZ_zYnmQaQxK0di6ZpUPEn6yyLu3VEaA

➔

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.

6.6 Servizio GMail

Il servizio *GMail* permette di leggere e inviare email con *GMail* 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 usare il servizio è sufficiente avere un account GMail. 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.
Download Full Email	(opzionale) Se impostata a true indica al metodo <i>ReadEmails</i> 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:

```
public void Form.SendEmail()
{
    MailServer ms = new()
    ms.UserEmailAddress = "v.marino@gmail.com"
    ms.UserPassword = "ju9fdjRt"
}
```

6.6.1 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. Per ottenere le mailbox è necessario usare la funzione *GetMailboxes* che riempie la collection *Mailboxes* dell'oggetto *MailServer* con oggetti di tipo *MailboxItem* che possiedono le seguenti proprietà:

Name	Nome della mailbox.
Last Read Email ID	È una stringa che rappresenta il numero progressivo dell'ultima email presente nella mailbox. Questa proprietà viene valorizzata quando si popola la collection <i>Mails</i> della mailbox.

Ogni oggetto di tipo *MailboxItem* può contenere altri oggetti dello stesso tipo e oggetti di tipo *MailItem* che rappresentano le email.

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 pro-

prietà *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.

Se sono stati scaricati solo gli headers delle email, per ottenere l'email completa occorre utilizzare la funzione *GetSingleEmail* 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.

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 = "ju9fdjR&"
    //
    // 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
        }
    }
    //
    //
}

```

6.6.2 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 = "ju9fdjR&"
    //
    // 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
        }
    }
}
```

6.6.3 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.

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*.

Dopo aver creato tutte le email che si vogliono mandare occorre salvare la collection in cui sono state aggiunte. Dopo l'invio le email saranno spostate in automatico nella mailbox *[Gmail]/Inviata*. Un esempio di codice è il seguente:

```

public void Form.SendEmail()
{
    MailServer ms = new()
    ms.UserEmailAddress = "v.marino@gmail.com"
    ms.UserPassword = "ju9fdjR%"
    //
    // 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])
}

```

6.7 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:

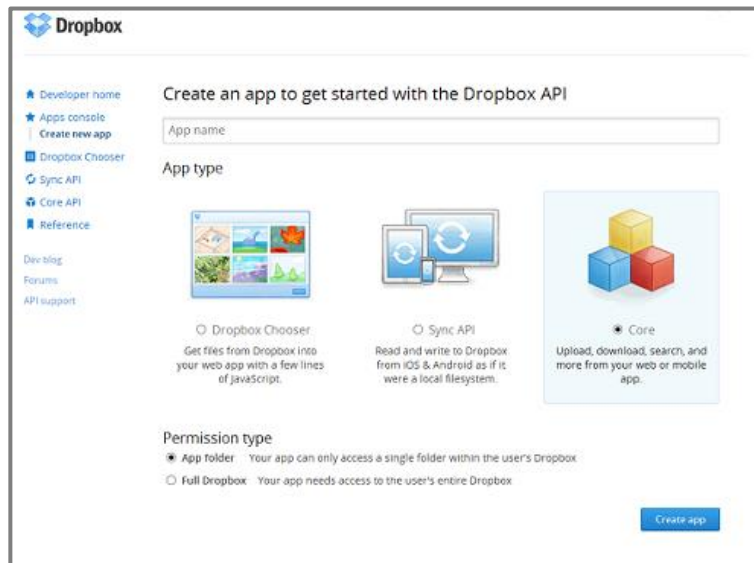
- creazione cartelle;
- lista dei file e directory presenti sul server;
- upload e download di file;
- eliminazione di file e cartelle;
- copia e spostamento di file da una cartella a un'altra.

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

6.7.1 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*.

6.7.2 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 <i>true</i> 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)
}

```

6.7.3 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 procedura *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)
    }
}

```

6.7.4 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 <i>File</i> nel caso in cui si stia caricando un file, <i>Directory</i> 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 la 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])
            }
        }
    }
}
```

6.7.5 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 = "chdsvbn7n8vfx8a"
    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])
                    }
                }
            }
        }
    }
}
```

6.7.6 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 = "fomkdSca0txacm3"
    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
        }
    }
}
```

6.7.7 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
    }
  }
}
}
}

```

6.7.8 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.

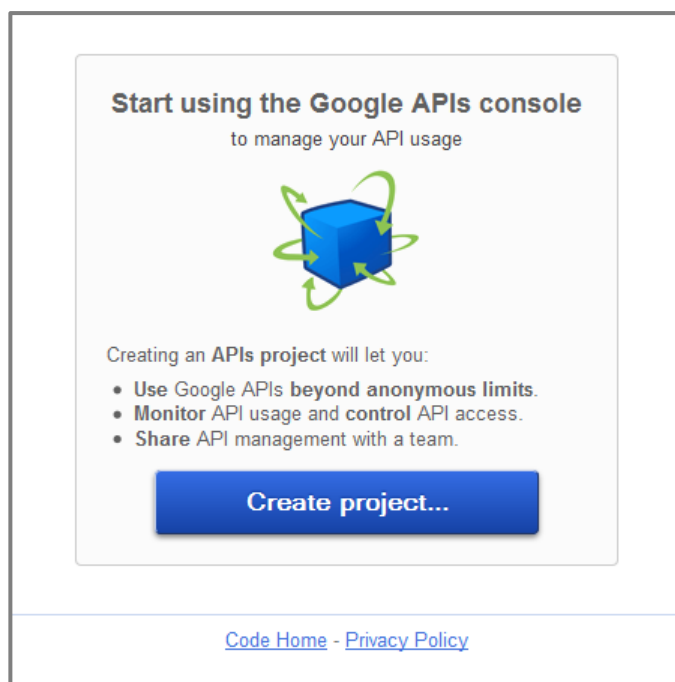
Per aggiungere un nuovo file occorre premere il bottone con la stella presente sopra la lista dei file e indicare qual è il file che si vuole caricare. Per eliminare un file occorre premere il pulsante con la x rossa presente nella stessa riga del file da cancellare. Per scaricare un file occorre premere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

6.8 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*.

6.8.1 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 ErrorMessage*.

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-20uhI_lkTDJzDF1K-2YKZ5DL1AXpHrpd1"
    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*.

6.9 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.

6.9.1 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 ErrorMessage*.

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 = "00000000539J984Z"
    ds.ClientSecret = "a5mWlekFMWxEdfVukUD1So9fVp2Sb32u"
    ds.AccessToken = "EwBAAQ1DBAAU1bRWyAJjK5w968Ru3Cyt/6GvwXwAAKZ5BI+WsUJajshIW3X-
    1XLpqaMFGb/d-8Eer8708S1dKeIMoA15kqeqjKyUD9w6dQ0WEfSb3fgY7mnFBgt17kFcJ83-
    BNxYElkiqVigaTrHHSIRvoxE-3jjeYkHKvcipAZk+PTGmQjHMSW0xHrYFEVVGonDp3eD1Yq-
    BB6f1ZEQOfYngb6+6Txav6F+h3zck0sJmjrk-CVe9G217E9Jfhd0jrj/Foed8Q7GtPwGCM-
    3K0GfTdsTITzushvU/puwP7rhCakOveM81NUIa9IbvRq1GQ8-yiXgmC/o7bWfa9WT3gopCx-
    16U+p00mCYRIvyrtT4PpMJhAwIB3HMr24dLZmdr3Bk2cDZgAACNLr1p0o4G9-ZEAHf5nsvf-
    RwgNrHgdzndCp9XJ5HDQ/9RvtPjTrhPeucSwh9CQZ0miAJyI1kJlwTkPreYdDh7rXYcbx/j-
    d-ug5pWRT+kthzIRy+78Zr0XXMbOi+pYNncvnbQ/6yz+CatEyXBFFca/y0t24kBS2enyjN/-
    ieeWib+qWHN3L-gUheTZHfLTqb2rjdj2Vs6Wtz4XmSONIWORb1S4LxfveeOafYTIJACMDRbO-
    OOHsu5g/y19XEDVeJ8DLTmRyJ-C4FJvzZzP7bEWLcz4B9F8YrwhYJtOJ9Kgo7bR/98vLPw-
    /7B64IaFr0w/IRK55Y/tijrjkoDnULQcZCm4d-8gH7HL11N/4kSKiyEW1LrtZQWcvCIui8N-
    kBvVuwAA"
    ds.RefreshToken = "ChQImD3v9KjczWfUen9wJS!2L7bukZE86n1jXF3cZEswCHB!cZ0e1XZKHD-
    mxqfDaSLerM6-LLBAXWq*8Hoag2qN54haJohA6I0oLH0S2hUMGiS45I703I3ByZojqD*kmp-
    GYkYI1gbZ1ZA0Y1gi*FhDXua-VMFv1c8jSC8ys!*jZuJ1bVuq7P5!VoDYASsCSRSTd2ME47J-
    8E6*WgVhuv1oAGOItCJ0jY9qHabmoLj1gTOS-uCRRGmdw!T9KXdrP0AcMXfkZRRB8n4e!E!-
    3zL51vt*!jq7diM!44lhZpP8sjK1EVBoycOx1PM1so7Ck4ol-Tr2ojIvjg4seWoJAO4j5Sd-
    n!UkIowmwp8aymuHKPWwLcVj1PhMUWqbC3WZhRj*VAWr7jVfvgbHstdFr1i-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*.

6.10 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.